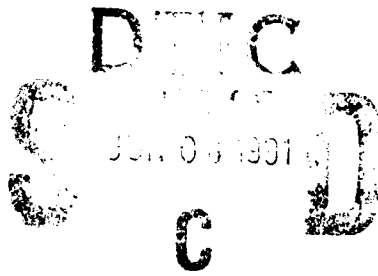


AD-A236 684



ADA COMPILER EVALUATION CAPABILITY

Version Description Document, Release 2.0

Thomas Leavitt
Kermit Terrell

Boeing Military Airplanes
Post Office Box 7730
Wichita KS

May 1991

Interim Report



Application For	
Full Text	☒
DTIC	☐
DTIC	☐
DTIC	☐
By	
Date	
A-1	
Dist	Special

Approved for public release; distribution unlimited.

AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

91 6 4 109

91-01209



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Service, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Project Room (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE May 1991	3. REPORT TYPE AND DATES COVERED Interim
----------------------------------	----------------------------	---

4. TITLE AND SUBTITLE Ada Compiler Evaluation Capability Version Description Document, Release 2.0	5. FUNDING NUMBERS C-F33615-86-C-1059 PE-63756D PR-2853 TA-01 WU-03
--	--

6. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Boeing Military Airplanes Post Office Box 7730 Wichita KS	7. PERFORMING ORGANIZATION REPORT NUMBER
---	--

8. AUTHOR(s) Raymond Szymanski (513) 255-3947 Avionics Directorate (WL/AAAF) Wright Laboratory Wright-Patterson, AFB, Oh 45433-6543	9. SPONSORING/MONITORING AGENCY REPORT NUMBER WL-TR-91-1039
---	--

10. DISTRIBUTION STATEMENT Approved for Public Release; Distribution is unlimited	11. DISTRIBUTION STATEMENT
--	----------------------------

The Ada Compiler Evaluation Capability (ACEC) is a set of over 1500 performance and usability tests used to assess the quality of Ada compilers. The ACEC also provides statistical analysis tools to assist in analyzing the results generated by the ACEC. The ACEC is documented through three major documents; the ACEC Reader's Guide, the ACEC User's Guide and the ACEC Version Description Document.

This document, the ACEC Version Description Document, records data pertinent to the status and usage of the ACEC. For each test, this document a) provides a terse description, b) identifies a source file, and c) identifies primary, secondary and incidental purposes.

12. SUBJECT TERMS Ada, Compiler, Evaluation, ACEC Metrics, Evaluation & Validation Project	13. NUMBER OF PAGES 286
14. SECURITY CLASSIFICATION Un	15. LIMITATION OF ABSTRACT DTIC users

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

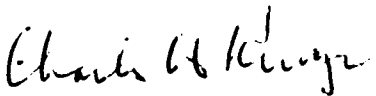
This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


RAYMOND SZYMANSKI
Project Engineer

25 March 1991
Date

FOR THE COMMANDER



CHARLES H. KRUEGER, JR.
Director
System Avionics Division
WPAFB Laboratory

3 APR 1991
Date

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify WL/AAAF, WPAFB, OH 45433-6543 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

BOEING

FSCM NO. 82918

THIS DOCUMENT IS:

CONTROLLED BY SOFTWARE AND LANGUAGES 75380

ALL REVISIONS TO THIS DOCUMENT SHALL BE APPROVED
BY THE ABOVE ORGANIZATION PRIOR TO RELEASE.

PREPARED UNDER ☒ CONTRACT NO. F33615-86-C-1059
☐ IR&D
☐ OTHER

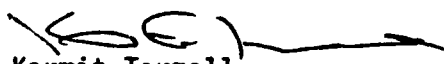
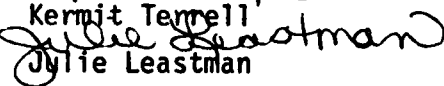
PREPARED ON FILED UNDER
DOCUMENT NO. D500-12472-1 MODEL

TITLE Ada COMPILER EVALUATION CAPABILITY (ACEC)
VERSION DESCRIPTION DOCUMENT
RELEASE 2.0

ORIGINAL RELEASE DATE May 4, 1990

ISSUE NO. TO DATE

ADDITIONAL LIMITATIONS IMPOSED ON THIS DOCUMENT
WILL BE FOUND ON A SEPARATE LIMITATIONS SHEET.

PREPARED BY		75380	4 May 90
CHECKED BY	 Julie Leastman	75380	4 May 1990
SUPERVISED BY	 Sam Ashby	75380	4 May 1990
APPROVED BY	 E. T. Startzman	75380	4 May 1990

SIGNATURE

ORGN

DATE

LIMITATIONS

This document is controlled by the Boeing Military Airplanes (BMA) Software and Languages Organization. All revisions to this document shall be approved by the above organization prior to release.

ABSTRACT

This document identifies and describes Version 2 of the Ada Compiler Evaluation Capability (ACEC). The Version Description Document (VDD) records data pertinent to the status and usage of the ACEC Software Product.

ACEC
Version Description Document (VDD)

Contents

1	SCOPE	5
1.1	IDENTIFICATION	5
1.2	PURPOSE	5
1.3	INTRODUCTION	5
2	REFERENCED DOCUMENTS	7
2.1	GOVERNMENT DOCUMENTS	7
2.2	NON-GOVERNMENT DOCUMENTS	7
3	VERSION DESCRIPTION	8
3.1	INVENTORY OF MATERIALS RELEASED	8
3.2	INVENTORY OF CSCI CONTENTS	8
3.3	ADAPTATION DATA	8
3.4	INTERFACE COMPATIBILITY	8
3.5	BIBLIOGRAPHY OF REFERENCE DOCUMENTS	9
3.6	INSTALLATION INSTRUCTIONS	9
3.7	POSSIBLE PROBLEMS AND KNOWN ERRORS	9
4	NOTES	10
4.1	ACRONYMS AND ABBREVIATIONS	10
5	APPENDICES	11
5.1	Appendix I, TEST PROBLEM DESCRIPTIONS	11
5.2	Appendix II, TEST PROBLEM TO SOURCE FILE MAP	123
5.3	Appendix III, TAPE DESCRIPTION	167
5.4	Appendix IV, QUARANTINED TEST PROBLEMS	174
5.5	Appendix V, ACEC KEYWORD INDEX - 1	190
5.6	Appendix VI, ACEC KEYWORD INDEX - 2	207
5.7	Appendix VII, SYSTEM DEPENDENT TEST PROBLEMS	270
5.8	Appendix VIII, OPTIMIZATION TECHNIQUES	273
5.9	Appendix IX, WITHDRAWN TEST PROBLEMS	283

1 SCOPE

1.1 IDENTIFICATION

This Version Description Document (VDD) describes Version 2 of the Software Product of the Ada Compiler Evaluation Capability (ACEC) System.

1.2 PURPOSE

The purpose of the ACEC is to provide a capability for quantitative evaluation of Ada compilation systems. The ACEC system is a set of software test programs and associated support tools and procedures which will determine the performance characteristics of Ada compilation systems. This includes the capability to automatically compare the results obtained on different Ada compilation systems. Such comparisons will isolate language constructions where one optimization has particular problems relative to other compilers tested. The ACEC will test for the presence of particular compiler optimizations.

ACEC software is comprised of the test suite and the support tools. For a list and description of the tests contained in the test suite, see Section 5.1, "Appendix I, Test Problem Descriptions." The support tools consist of:

- **INCLUDE** — A tool to perform source text inclusion. It will assist in adapting programs to particular targets.
- **FORMAT** — A tool to extract the timing and code expansion data which the execution of the test suite wrote to standard output in a human readable form.
- **MED_DATA_CONSTRUCTOR** — A tool to convert the output from various runs of **FORMAT** (on different systems) into a form usable by the **MEDIAN** program and the **SSA** program. This format is one of two initialized array aggregates: one which identifies each test problem (by name) and gives the execution time for the problem; and one which identifies each test problem (by name) and gives the code expansion size for the problem.
- **MEDIAN** — A tool to compare results of performance tests of various systems.
- **SINGLE SYSTEM ANALYSIS (SSA)** — A tool to analyze the results of related sets of performance tests from a single system.

1.3 INTRODUCTION

This document describes the ACEC Software Product as contained on the release tape. It describes the compilation units, programs, test problems, and sample data contained on the

distribution tape. This document contains several appendices with release dependent information, making the Reader's Guide and User's Guide insensitive to releases. See the following table for a brief description of each of the appendices included in this document.

Appendix	Name	Contents
I	Test Problem Descriptions	List of test problem names with a brief description of each. New or withdrawn tests are identified.
II	Test Problem to Source File Map	List of test problems and the source file they are contained in.
III	Tape Description	List of files on the delivery tape
IV	Quarantined Test Problems	Cross reference of test problems observed to fail on some systems
V	ACEC Keyword Index - 1	List of primary purposes (with LRM references) and their associated test problems, as well as secondary, and incidental purposes, and comparison tests.
VI	ACEC Keyword Index - 2	List of test problems with their primary purposes (which may be for comparison with other tests).
VII	System Dependent Test Problems	List of test problems which exercise system dependent features.
VIII	Optimization Techniques	List of optimization techniques and the benchmarks designed to test them
IX	Withdrawn Test Problems	List of test problems which have been withdrawn

2 REFERENCED DOCUMENTS

The following documents are referenced in this VDD.

2.1 GOVERNMENT DOCUMENTS

MIL-STD-1815A	Reference Manual for the Ada Programming Language
---------------	---

2.2 NON-GOVERNMENT DOCUMENTS

D500-12470-1	Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) User's Guide RELEASE 2.0 Boeing Military Airplanes P. O. Box 7730 Wichita, Kansas
--------------	--

D500-12471-1	Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide RELEASE 2.0 Boeing Military Airplanes
--------------	---

3 VERSION DESCRIPTION

For the second release of the ACEC Software Product, this section contains information on the inventory of materials released, the inventory of CSCI contents, the adaptation data (where applicable), interface compatibility (where applicable), bibliography of reference documents, installation instructions, and possible problems and known errors.

3.1 INVENTORY OF MATERIALS RELEASED

The release of the Software Product of the ACEC will be comprised of:

- the distribution tape,
- the User's Guide,
- the Reader's Guide, and
- this VDD.

3.2 INVENTORY OF CSCI CONTENTS

The ACEC Software Product consists of two CSCIs: the Operational Software (test suite) and the Support Software (support tools). The distribution tape contains the test suite and the support tools. For a list of the contents of the test suite, see Section 5.1, "Appendix I, Test Problem Descriptions." The five support tools are INCLUDE, FORMAT, MED DATA CONSTRUCTOR, MEDIAN, and SSA. A brief description of each is found in Section 1.2, "PURPOSE" of this document. For more detailed information, refer to the User's Guide, Sections: "USING INCLUDE", "PREPARING THE DATA", "RUNNING MEDIAN", and "SSA".

Refer to Section 5.3, "Appendix III, TAPE DESCRIPTION" for a listing of the files on the distribution tape.

3.3 ADAPTATION DATA

The ACEC Software Product has no "unique-to-site" data. Appendix VII (System Dependent Test Problems) identifies all system dependencies contained in the items being released.

3.4 INTERFACE COMPATIBILITY

Not applicable. For information on how the test suite and the support tools interface, refer to the User's Guide, Section "PREPARING THE DATA".

3.5 BIBLIOGRAPHY OF REFERENCE DOCUMENTS

Refer to Section 2, "REFERENCED DOCUMENTS."

3.6 INSTALLATION INSTRUCTIONS

For information detailing how to install and checkout the delivered ACEC Software Product, refer to the User's Guide, Section "INSTALLATION".

3.7 POSSIBLE PROBLEMS AND KNOWN ERRORS

Refer to 5.4 Appendix IV, "Quarantined Test Problems."

4 NOTES

This section contains information only and is not contractually binding.

4.1 ACRONYMS AND ABBREVIATIONS

ACEC	Ada Compiler Evaluation Capability
BMA	Boeing Military Airplanes
CSCI	Computer Software Configuration Item
LRM	Language Reference Manual, specifically Reference Manual for the Ada Programming Language, MIL-STD-1815A
SSA	Single System Analysis
TOR	Technical Operating Report
VDD	Version Description Document

5 APPENDICES

5.1 Appendix I, TEST PROBLEM DESCRIPTIONS

This appendix contains an alphabetical list of test problem names with a brief description of each.

Problem Test Name	Problem Test Description
a_star	Implementation of an Artificial Intelligence programming technique.
acker1	Classical test, Ackermann's function, suppression of pragmas; intensive test of function calling
acker2	Classical test, Ackermann's function, no suppression intensive test of function calling
activation1	Use 'address attribute and procedure calling to measure the size of activation record when calling a procedure in a separate package. Compare with activation2, which calls on an INLINED subprogram in the same compilation unit.
activation2	Use the 'address attribute to measure the size of the activation record of inlined subprogram. Compare with activation1, not specified as inline.
ai_create_delete_kb	AI application study. Large example program using non-numeric processing. Create and Delete KB
ai create object	AI application study. Large example program using non-numeric processing. Create objects, with different <i>degrees of inheritance of attributes from higher level conceptual objects.</i>
ai_load_kb_from_file	AI application study. Large example program using non-numeric processing. Loads a Knowledge_Base from a file and then deletes it.
ai modify object	AI application study. Large example program using non-numeric processing. Modify values of attributes within objects.
ai_query	AI application study. Large example program using non-numeric processing. Query the relationships between objects.

Problem Test Name	Problem Test Description
alias1	This is a problem which appears to be subject to loop invariant motion but is not because the object has aliases (allocated object referenced by two different) access types. It includes an error check.
alias2	This is a problem which appears to be subject to common subexpression elimination and invariant motion but is not because the object has aliases (allocated object referenced by two different) access types. It includes an error check.
alias3	This is a problem which appears to be subject to dead assignment elimination but is not because the object has aliases (allocated object referenced by two different) access types. It includes an error check.
alias4	This is a problem which appears to be subject to folding but is not because the object has aliases (allocated object referenced by two different) access types. It includes an error check.
alias5	This is a problem which is subject to loop invariant motion.
alias5x	This is a problem which has had loop invariant motion performed "by hand."
alias6	This is a problem which is subject to common subexpression elimination.
alias6x	This is a version of alias6 where common subexpression elimination has been performed "by hand."
alias7	This is a problem which is subject to dead assignment elimination.
alias7x	This is a version of alias7 where the dead assignment has been eliminated "by hand."
alias8	This is a problem which is subject to folding.
alias8x	This is a version of alias8 folded "by hand."
alias9	This is a problem which is subject to loop invariant motion.

Problem Test Name	Problem Test Description
alias10	This is a problem which is subject to common subexpression elimination.
alias11	This is a problem which is subject to dead assignment elimination.
alias12	This is a problem which is subject to folding.
alias13	This is a problem which appears to be subject to loop invariant motion but is not because the object has aliases. It includes an error check.
alias14	This is a problem which appears to be subject to common subexpression elimination but is not because the object has aliases. It includes an error check.
alias15	This is a problem which appears to be subject to dead assignment elimination but is not because the object has aliases. It includes an error check.
alias16	This is a problem which appears to be subject to folding but is not because the object has aliases. It includes an error check.
arti_asum	Avionics application study example; Angle sum
arti_atan2	Avionics application study example, Arctangent
arti_cos	Avionics application study example; Cosine
arti_fmod	Avionics application study example, Float mod
arti_ifpm_control	Avionics application study example; get input data, initialize In Flight Performance Monitor (IFPM); check rotor check problem; calls on IFPM_IO; IFPM_INIT; IFPM_ROTORS;
arti_ifpm_init	Avionics application study example; initialize In Flight Performance Monitor (IFPM) system
arti_ifpm_io	Avionics application study example Inflight Performance Monitor (IFPM), I/O operation
arti_ifpm_rotors	Avionics application study example IFPM check rotor performance

Problem Test Name	Problem Test Description
arti_nairini	Avionics application study example; Initialize navigation
arti_nscni	Avionics application study example; Initialize data for Navigation Prime Data Calculation Routine (NSCNDV)
arti_nutmini	Avionics application study example; Initialize data for NUTMCON and call it. NUTMCOM is a UTM conversion routine.
arti sin	Avionics application study example, Sine
async1	Test of console output and task scheduling. It has two tasks: a high priority task which does 100 PUTs to the console and a low priority task which set a boolean flag if it ever gets control. After completion of the high priority tasks, the state of the flag will let the system know whether the low priority task ever got control. The test problem monitors if the low priority task got control: never, sometimes, or always.
async2	Perform task rendezvouses while the system has a task waiting on a GET from a console in another task.
async3	Perform 10 000 procedure calls another task is waiting in a long DELAY loop. Need to perform a fairly long test problem to permit possible overheads from polling to show up.
async4	Perform 10 000 procedure calls while another task is waiting on a terminal GET. Need to perform a fairly long test problem to permit possible overheads from polling to show up.

Problem Test Name	Problem Test Description
async5	Test of asynchronous I/O. It has two tasks: a high priority task which does 1000 READs from direct file and a low priority task which set a boolean flag if it ever gets control. After completion of the high priority task, the state of the flag will let the system know whether the low priority task ever got control. The test problem monitors if the low priority task got control: never, sometimes, or always.
auto	Classical test, from the Computer Family Architecture (CFA) study; autocorrelation program
avl_0	An AVL tree, named after (Adelson-Velskii and Landis) is a balanced tree supporting both keyed and positional records in a random order and delete them in reverse insertion order. Tree size is 100 records.
avl 1	An AVL tree, This problem inserts records in ascending key order and deletes them in descending key order. Tree size is 100 records.
avl_2	An AVL tree test, it attempts to insert a duplicate for each key (which should fail). Tree size is 100 records.
avl 3	An AVL tree, this problem searches for each record by key. It will fail if the records are not found. Tree size is 100 records.
avl_4	An AVL tree, this problem searches for each record by position. It will fail if the proper record is not found. Tree size is 100 records.
avl 5	An AVL tree test, this problem searches for keys not present in tree but close to contained values. It will fail if the records are found. Tree size is 100 records.

Problem Test Name	Problem Test Description
avl_6	An AVL tree, named after (Adelson-Velskii and Landis) is a balanced tree supporting both keyed and positional records in a random order and delete them in reverse insertion order. Tree size is 1_000 records.
avl_7	An AVL tree, This problem inserts records in ascending key order and deletes them in descending key order. Tree size is 1_000 records.
avl_8	An AVL tree test, it attempts to insert a duplicate for each key (which should fail). Tree size is 1_000 records.
avl_9	An AVL tree, this problem searches for each record by key. It will fail if the records are not found. Tree size is 1_000 records.
avl_10	An AVL tree, this problem searches for each record by position. It will fail if the proper record is not found. Tree size is 1_000 records.
avl_11	An AVL tree test, this problem searches for keys not present in tree but close to contained values. It will fail if the records are not found. Tree size is 1_000 records.
bmt	Classical test, from CFA study - boolean matrix transpose heavy use of 2-D array references
bsort1	Classical test, sort program, variant of quicksort which exploits partially sorted sequences. This sequence is not sorted.
bsort2	Classical test, sort program, variant of quicksort which exploits partially sorted sequences. This sequence is sorted.

Problem Test Name	Problem Test Description
cat1	LRM requires that assignment not modify the left hand side if the expression on the right hand side will raise an exception. In general, a compiler can comply with this requirement by copying intermediate results of catenation operators to a temporary copy and moving it into the left hand side after verifying that no exceptions were raised. In some cases, an optimizing compiler will be able to verify at compile time that no exceptions can be raised and can generate faster code by copying directly into the left hand variable without using a temporary. This set of problems will explore options in this areas. CAT1 can be easily verified not to raise length errors.
cat2	Version of cat1 which will require runtime length check
cat3	Version of cat1 which will raise CONSTRAINT_ERROR
cio1	Test performance of console output. Test problem redisplay the same string to permit ACEC user to observe if the system has an optimizing screen manager which omits transmitting characters which do not change
cio2	Test performance of console output. Test problem displays string with different characters in each position, forcing an explicit transmission of each character.
cio3	Test performance of console output. Test problem displays string followed by display of a blank string. Many terminals have control commands which permit the efficient deletion (blanking) of an entire line.

Problem Test Name	Problem Test Description
cio4	Test performance of console output. Test problem displays string followed by a display of a string with every other character changed. Some sequence of terminal commands might include positioning command to move cursor over each unchanged character, but the time to process these commands in this example can easily be larger than a simple delete and transmission of entire string. This should not be slower than cio2
cio5	Test performance of console output. Test problem changes one character in the display string from the prior line. Some sequence of terminal commands can move one cursor position instruction, a delete, and an insert, which can be much faster than a general redisplay where every character must be transmitted.
cio6	Test performance of console output. Test problem inserts 6 characters in the middle of a display and deletes 6 at the end of the display. Some terminals can perform this update with a few commands.
cio7	Test performance of console output. Test problem deletes 6 characters in the middle of a display string. Some terminals can perform this update with one or two commands (position and delete).
cio8	Test performance of console output. Test problem deletes 50 characters from tail of a display string. Some terminals can perform this update with one or two commands (position and delete).
cio9	Test performance of console output. Test problem adds 50 characters to the tail of a 20 character display string. Some terminals can perform this update with without transmitting each leading character.

Problem Test Name	Problem Test Description
cio10	Test performance of console output. Test problem replaces string with 70 non blank characters with one non-blank at position 60. Some terminals can perform this update quicker by blanking line, positioning cursor, and inserting one character than by transmitting all the (blank) characters.
cio11	Test performance of console output. Test problem replaces string with 70 non blank characters with first 20 characters blank.
cio12	Test performance of console output. Test problem replaces string with 70 non blank characters with first 20 characters different.
cio13	Test performance of console output. Test problem replaces string with 2 character string with another 2 character string.
cio14	Test performance of console output. Test problem replaces string with 60 blanks and an X to all blank string.
cqsort	Classical test, quicksort. integer array, no suppression
claim01	Test function returning a discriminated record which contains variable sized components. This might expose runtime systems which allocate and not reclaim the space.
claim02	recursive function returning an unconstrained STRING type. This might expose runtime systems which allocate and not reclaim the space.
claim03	Test function returning a discriminated record which contains one variable sized component. This might expose runtime systems which allocate and not reclaim the space.

Problem Test Name	Problem Test Description
claim04	Test function returning a discriminated record which contains multiple discriminated records. This might expose runtime systems which allocate and not reclaim the space.
claim05	Test function returns a "large" (a 1000 character array) fixed size type. This might expose runtime systems which allocate and not reclaim the space.
claim06	Test function returns a "small" (a 20 character array) fixed size type. This might expose runtime systems which allocate and not reclaim the space.
claim07	Procedure which declares an unconstrained type as a private type within a nested generic package. This might allocate and not reclaim storage.
claim08	Package with an unconstrained array type (STRING) as as a private type within a generic package. Actual formal parameter instantiates with length 8. This might allocate and not reclaim storage.
claim09	Use a GOTO to exit a nested FOR loop. This might implicitly allocate space which is not reclaimed.
claim10	Use a GOTO to exit from nested DECLARE blocks. This might implicitly allocate space which is not reclaimed.
claim11	Raise on exception to exit from a nested FOR loop. This might implicitly allocate space which is not reclaimed.
claim12	Raise an exception to exit from nested DECLARE blocks. This might implicitly allocate space which is not reclaimed.

Problem Test Name	Problem Test Description
claim13	Raise an exception to exit from nested DECLARE blocks where the blocks have exception handlers. This might implicitly allocate space which is not reclaimed.
claim14	Expression containing two function calls on a function with side effects returning an unconstrained type. The first time it is called it will return a string, the second time it will raise a predefined exception by dividing by zero.
claim15	Expression containing two function calls on a function with side effects returning an unconstrained type. The first time it is called it will return a string, the second time it will raise a user defined exception.
claim16	Problem declaring nested discriminated objects within a nested package. The concern is that the system might allocate space and not reclaim it.
claim17	Problem which uses the 'IMAGE attribute. This might cause some systems to allocate and not reclaim space. 100,000 executions per timing loop iteration.
claim18	Boolean operator on arrays requiring temporaries. Some systems may allocate and not reclaim space here.
claim19	Boolean operator on arrays requiring temporaries where an exception is raised part way through the evaluation. Some systems may allocate and not reclaim space here.
claim20	Catenation of array slices where temporaries are required. Some system may allocate and not reclaim space.
claim21	Catenation of array slices where temporaries are required where an exception will be raised by the execution of the problem. Some system may allocate and not reclaim space.

Problem Test Name	Problem Test Description
claim22	Catenation of array slices where temporaries are required to save the results of function calls and where an exception will be raised. Some system may allocate and not reclaim the space.
claim23	Problem performs I/O to and from a string. Some systems may allocate and not reclaim the space.
claim24	Problem performs I/O to and from a file. Some systems may allocate and not reclaim the space.
claim25	Problem instantiates enumeration.io multiple times. Some systems might allocate and not reclaim space.
claim26	Problem which allocates a record with a dynamically sized component and explicitly deallocates the record. 100,000 executions per timing loop iteration.
claim27	Problem which allocates a record with a dynamically sized component and implicitly deallocates it. 100,000 executions per timing loop iteration.
claim28	Problem which performs a SELECT with multiple open alternatives within a declare block.
claim29	Problem which performs a SELECT with multiple open alternatives where the SELECT is not within a declare or subprogram defining an exception handling frame. This is similar to CLAIM28 except that some systems might not reclaim space for SELECT temporaries until a frame is exited.
claim30	Problem which performs a SELECT with DELAY alternatives where no alternative is immediately open but where alternative is satisfied before DELAY expires. The concern is that the system DELAY queue might fill up with dummy entries and the space used not be reclaimed.

Problem Test Name	Problem Test Description
claim31	Problem which performs a SELECT with DELAY alternatives where no alternative is immediately open but where alternative is satisfied before DELAY expires. The concern is that the system DELAY queue might fill up with dummy entries and the space used not reclaimed. This is similar to CLAIM30 except that the entry call loop does NOT contain an exception handling frame.
claim32	Problem which performs a timed entry call which is not immediately callable. Because of task priorities, a DELAY alternative is taken, and the lower priority task should then be scheduled and become callable. The concern is that the system DELAY queue might fill up with dummy entries and the space not be reclaimed.
claim33	Problem which performs a timed entry call which is not immediately callable. Because of task priorities, a DELAY alternative is taken, and the lower priority task should then be scheduled and become callable. The concern is that the system DELAY queue might fill up with dummy entries and the space not be reclaimed.
claim34	Different from claim32 in that the loop performing entry calls is NOT in an exception handling frame. Create a task which immediately terminates. The concern is that the system may allocate space for task control blocks and not reclaim it.
claim35	Create a task which immediately terminates. The concern is that the system may allocate space for task control blocks and not reclaim it.
claim36	Problem which create a task and aborts it. The concern is that the system may allocate space for task control blocks and not reclaim it.

Problem Test Name	Problem Test Description
claim37	Problem which create a task and aborts it. The concern is that the system may allocate space for task control blocks and not reclaim it. This differs from claim36 in that the loop repetitively creating and aborting tasks does not contain an exception handler.
claim38	Problem elaborates a declarative region including assigning to object in a collection before calling on a function which raises an exception before leaving the declarative region.
claim39	Problem calls on function CLOCK. Some system may allocate and not reclaim space.
claim40	Test function returning a discriminated record which contains variable sized components. This might expose runtime systems which allocate and not reclaim the space. This is a version of claim01 with function specified as INLINE.
claim41	recursive function returning an unconstrained STRING type. This might expose runtime systems which allocate and not reclaim the space. This is a version of CLAIM02 with function specified as INLINE.
claim42	Test function returning a discriminated record which contains one variable sized component. This might expose runtime systems which allocate and not reclaim the space. This is a version of claim03 with function specified as INLINE.
claim43	Test function returning a discriminated record which contains multiple discriminated records. This might expose runtime systems which allocate and not reclaim the space. This is a version of claim04 with function specified as INLINE.

Problem Test Name	Problem Test Description
claim44	Test function returns a "large" (1000 character array) fixed size type. This might expose runtime systems which allocate and not reclaim the space. This is a version of claim05 with function specified as INLINE.
claim45	Test function returns a "small" (20 character array) fixed size type. This might expose runtime systems which allocate and not reclaim the space. This is a version of claim06 with function specified as INLINE.
claim46	Expression containing two function calls on a function with side effects returning an unconstrained type. The first time it is called it will return a string, the second time it will raise a predefined exception by dividing by zero. This is a version of claim14 with function specified as INLINE.
claim47	Expression containing two function calls on a function with side effects returning an unconstrained type. The first time it is called it will return a string, the second time it will raise a user defined exception. This is a version of claim15 with function specified INLINE.
common	Optimization test constructed with many common subexpressions
complex_record01	Copy a field in a nested record definition to a simple variable. Variable is a packed enumeration type with a T'SIZE clause specified.
complex_record02	Copy a simple variable to a field in a nested record definition.
complex_record03	Reference a field in a nested record definition in an expression.

Problem Test Name	Problem Test Description
complex_record04	Pass a field in a nested record definition as an actual parameter to an IN OUT mode parameter.
complex_record05	Copy a nested variant record to another record.
complex_record06	Compare variant records, one of which is nested within another record definition.
complex_record07	Copy a field which is itself a record.
complex_record08	Pass a field which is a record embedded in a higher order record definition as an IN mode parameter to a subprogram.
complex_record09	Pass a field which is a record embedded in a higher order record definition as an IN OUT mode parameter to a procedure.
consistent1	This sequence of tests executes an IF statement which evaluates to FALSE, skipping the THEN part. Various numbers of statements are included in the THEN part to observe if the execution time is constant, as it would be if the condition always generated the same code to evaluate the condition and branches around the THEN clause and then this code always executes in the same amount of time. Differences can be caused by length of the branch required to skip over the THEN clause, or by alignment of the instructions. This problem has a simple procedure call in the THEN clause.
consistent2	This is one of the CONSISTENT family of problems. It has two simple procedure calls in the THEN clause.
consistent3	This is one of the CONSISTENT family of problems. It has three simple procedure calls in the THEN clause.
consistent4	This is one of the CONSISTENT family of problems. It has four simple procedure calls in the THEN clause.
consistent5	This is one of the CONSISTENT family of problems. It has five simple procedure calls in the THEN clause.
consistent6	This is one of the CONSISTENT family of problems. It has 100 simple procedure calls in the THEN clause.

Problem Test Name	Problem Test Description
consistent7	This is one of the CONSISTENT family of problems. It has a declare block with 100 procedure calls in the THEN clause.
crc0	Initializes cyclic redundancy check lookup tables. Test will fail if unique numbers are not generated.
crc1	Perform a CRC calculation on a string of length one. Test will fail if the values generated for the CRC check bytes are not those expected.
crc2	Perform a CRC calculation on a string of length 100. Test will fail if the values generated for the CRC check bytes are not those expected.
crc3	Perform a CRC calculation on a string of length 200. Test will fail if the values generated for the CRC check bytes are not those expected.
crc4	Perform a CRC calculation on a string of length 400. Test will fail if the values generated for the CRC check bytes are not those expected.
cse1	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE1 uses local variables with intervening procedure calls. It can be validly optimized.
cse2	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE2 evaluates the expression once and saves it in a local variable. If the time for CSE1 and CSE2 are comparable, then the system is doing common subexpression elimination.
cse3	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE3 evaluates the expression without intervening subprogram calls. A system might recognize these as common but not CSE1.

Problem Test Name	Problem Test Description
cse4	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE4 uses global variables and is NOT really common. If the time for it is equal to CSE1, then common subexpressions are NOT recognized across subprogram calls.
cse5	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE1 uses local variables with intervening procedure calls. The subscript expression (i, i, i) is common and references to it can be validly optimized.
cse6	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE6 evaluates the expression once and saves it in a local variable. If the time for CSE5 and CSE6 are comparable, then the system is doing common subexpression elimination.
cse7	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE7 evaluates the expression without intervening subprogram calls. A system might recognize these as common but not CSE5.
cse8	Set of tests to see if compiler recognized common subexpressions across subprogram calls. CSE8 uses global variables and is NOT really common. If the time for it is equal to CSE5, then common subexpressions are NOT recognized across subprogram calls.
cse9	Common subexpression separated by conditional procedure call. Saving and restoring register values is possible but may not be profitable.
cse10	Common subexpression using global separated by conditional procedure call.

Problem Test Name	Problem Test Description
d library 1	This set of test problems measure package elaboration times, in particular library package elaboration time for packages which declare objects of non-static sizes. This problem contains a declare block defining a type, no objects, and a null body. It can be translated into a null.
d.library.2	This is one of the d.library.* family of test problems. This problem contains a declare block defining a type, no objects, and a body with one procedure call.
d library 3	This is one of the d library.* family of test problems. This problem contains nested package declarations defining 4 dynamically sized arrays. Package body contains one procedure call.
d.library.5	This is one of the d.library.* family of test problems. A declare block which allocates 4 dynamically sized arrays on a heap in a named collection and deallocates them via UNCHECKED_DEALLOCATION. The allocation is expected to be roughly comparable to the overheads of allocating dynamically sized objects in a library package.
d library 6	This is one of the d library.* family of test problems. A declare block which allocates 4 dynamically sized arrays on a heap in a named collection and reclaims space by exiting the block the collection is declared in. The allocation is expected to be roughly comparable to the overheads of allocating dynamically sized objects in a library package.

Problem Test Name	Problem Test Description
d library 7	This is one of the d library * family of test problems. Measurement of time to elaborate 5 library packages which allocate 4 dynamically sized arrays (same 4 as used in the other problems in this set). The measurement of this problem has coarser error bounds because it must use a variation of the timing loop code since it is not possible to force the system to elaborate a library package more than once per program execution.
d.library.8	This is one of the d.library.* family of test problems. This problem is a versions of d.library.7 using nested packages rather than library packages. It can be much faster than d library 7 because a simpler (stack based) storage allocation scheme can be used for nested packages.
dead	Optimization test; constructed so that systems which do dead assignment elimination will do well
delay1	Language feature test, delay statements contending tasks, "DELAY 0.0;" All the DELAY problems are given an error code as they will be ignored by MEDIAN. DELAY problems do not follow the basic ACEC modeling assumption that the execution time for a problem is approximately the product of a system factor and a problem factor. The execution time for a (long) DELAY statement should be roughly constant and implementation independent - a fast system should not execute a DELAY 1.0; faster than one second! If measurements are made using CPU time, then the modeling assumption would apply, but that is not expected to be the primary mode of operations.
delay2	Language feature test, delay statements contending tasks, delay 0.000001;

Problem Test Name	Problem Test Description
delay3	Language feature test, delay statements contending tasks, delay 0.000010;
delay4	Language feature test, delay statements contending tasks, delay 0.000100;
delay5	Language feature test, delay statements contending tasks, delay 0.001000;
delay6	Language feature test, delay statements contending tasks, delay 0.0100000;
delay7	Language feature test, delay statements contending tasks, delay 0.1000000;
delay8	Language feature test, delay statements no contending tasks, delay 0.000000;
delay9	Language feature test, delay statements no contending tasks, delay 0.000001;
delay10	Language feature test, delay statements no contending tasks, delay 0.000010;
delay11	Language feature test, delay statements no contending tasks, delay 0.000100;
delay12	Language feature test, delay statements no contending tasks, delay 0.001000;
delay13	Language feature test, delay statements no contending tasks, delay 0.010000;
delay14	Language feature test, delay statements no contending tasks, delay 0.100000;
delay_abort1	Tests whether the target system terminates an abnormal task at a DELAY 0.0 statement. If the system does not print an execution time error code for this statement, the system is properly terminating the aborted tasks before the DELAY 0.0 statement finishes.
delay_abort2	Test whether a system immediately kill a task which is aborted while in the middle of a DELAY or whether it waits until the DELAY completes.

Problem Test Name	Problem Test Description
delay zero0	One of a set of test problems to determine how a system treats the interaction of task switching and delay statements. In all problems in this set, there are two equal priority tasks which invoke a routine CHECKIN, passing the identification of the calling task, which detects whether there has been a task switch since the last time CHECKIN was called. This version implements CHECKIN as task entry, updating the task switch counter and old task identification within a rendezvous. It can detect whether the system is using a time-slice or a run-till-blocked task scheduling algorithm.

Problem Test Name	Problem Test Description
delay_zero1	One of a set of test problems to determine how a system treats the interaction of task switching and delay statements. In all problems in this set, there are the two equal priority tasks which invoke a routine CHECKIN, passing the identification of the calling task, which detects whether there has been a task switch since the last time CHECKIN was called. This version implements CHECKIN as a procedure, has no DELAY statements between calls on CHECKIN, and assumes that the user has requested a time-sliced task scheduling algorithm.
delay_zero2	One of the DELAY_ZERO family of test problems. This version implements CHECKIN as a procedure which updates the task switch counter and old task identification. This problem requests time-sliced task scheduling and inserts a literal DELAY 0.0 between calls on CHECKIN.
delay_zero3	One of the DELAY_ZERO family of test problems. This version implements CHECKIN as a procedure. It requests time-sliced task scheduling and inserts a DELAY DURATION_ZERO between calls on CHECKIN.

Problem Test Name	Problem Test Description
delay_zero4	One of the DELAY_ZERO family of test problems. This version implements CHECKIN as a procedure. It requests run-till-blocked task scheduling and inserts no DELAY statement between calls on CHECKIN.
delay_zero5	One of the DELAY_ZERO family of test problems. This version implements CHECKIN as a procedure. It requests run-till-blocked task scheduling and inserts a literal DELAY 0.0; statement between calls on CHECKIN.
delay_zero6	One of the DELAY_ZERO family of test problems. This version implements CHECKIN as a procedure. It requests run-till-blocked task scheduling and inserts a DELAY DURATION_ZERO; statement between calls on CHECKIN.
delay_zero6x	One of the DELAY_ZERO family of test problems. This version implements CHECKIN as a procedure. It uses system default task scheduling and inserts a DELAY DURATION_ZERO; statement between calls on CHECKIN.
delay_zero7	One of the DELAY_ZERO family of test problems. This version executes a DELAY DURATION_ZERO; with one other task in the system waiting on the delay queue.
delay_zero8	One of the DELAY_ZERO family of test problems. This version executes a DELAY DURATION_ZERO; with two other tasks in the system waiting on the delay queue.

Problem Test Name	Problem Test Description
des1	The Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Problem initializes data structures and calls on DES twice to encrypt and then decrypt a message.
des2	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 2 uses unpacked boolean arrays, indexing operations.
des3	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 3 moves constant array declarations to global scope; uses unpacked boolean arrays, indexing operations.
des4	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 4 separates initialization into a separate procedure. Uses unpacked boolean arrays, indexing operations. This (DES4) includes both initialize and call on the DES to encrypt and decrypt a message.
des4a	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 4 separates initialization into a separate procedure. Uses unpacked boolean arrays, indexing operations. This (DES4a) is the initialization procedure by itself.
des5	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 5 is similar to version 4 using packed arrays. DES5a is the setup procedure.
des5a	Data Encryption Standard (DES) algorithm. Version 5 is similar to version 4 using packed arrays. DES5a is the setup procedure.

Problem Test Name	Problem Test Description
des6	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 6 uses logical operators of unconstrained packed boolean array types to set bits. Has separate initialization procedure. This problem (DES6) includes initialization and two calls on DES, one to encrypt and one to decrypt.
des6a	Data Encryption Standard (DES) algorithm. Version 6 uses logical operators of unconstrained packed boolean array types to set bits. DES6a is the initialization procedure.
des7	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 7 uses constrained types. This (DES7) calls on an initialization procedure and two calls on DES to encrypt and decrypt a message.
des7a	Data Encryption Standard (DES) algorithm is used as an example of a nonnumeric application. Version 7 uses constrained types. This (DES7) calls on an initialization procedure and two calls on DES to encrypt and decrypt a message.
dhrys1_mod	Classical test, synthetic benchmark, Dhrystone without suppression.
dhrys2_mod	Classical test, synthetic benchmark, Dhrystone with suppression.
dhrys3_mod	Classical test, synthetic benchmark, Dhrystone with suppression and PRAGMA OPTIMIZE(SPACE)

Problem Test Name	Problem Test Description
elab1	These sets of tests examine system performance on test problems (compiled without suppression) which contain calls on subprograms declared in external procedures. They are designed to test if the system optimized some (any) of the elaboration checking code required to verify that the package body has been elaborated before subprograms in it are called. Because the calls are in a conditional statement, an optimizing compiler cannot move the elaboration check code out of the timing loop. This problem calls one procedure five times in a row. A good compiler would only generate code to perform one test.
elab10	Version of elab5 which specified suppression of predefined constraint checking.
elab2	This is one of the elab* set of test problems. This problem calls different procedures defined in different packages five times in a row. Because the calls are in a conditional statement, an optimizing compiler cannot move the elaboration check code out of the timing loop. Relative to ELAB1, this version must include 5 separate pieces of checking code.
elab3	This is one of the elab* set of test problems. This problem calls the same procedure four times in a row unconditionally and then tests the condition and calls it one more time. An optimizing compiler could move the elaboration checking code out of the timing loop.
elab4	This is one of the elab* set of test problems. This problem conditionally calls on five different procedures defined in the same package. Similar to ELAB1, an optimizing compiler could share the checking code for the package body elaboration.

Problem Test Name	Problem Test Description
elab5	This problem calls one procedure five times in a row within a conditional statement. The procedure is in a package which has a PRAGMA ELABORATE specified, permitting an optimizing compiler to omit checking code because the PRAGMA guarantees prior elaboration.
elab6	Version of elab1 which specified suppression of predefined constraint checking.
elab7	Version of elab2 which specified suppression of predefined constraint checking.
elab8	Version of elab3 which specified suppression of predefined constraint checking.
elab9	Version of elab4 which specified suppression of predefined constraint checking.
enum io1	Instantiate enumeration io in declare block, is amenable to loop invariant motion.
enum io2	Instantiate enumeration io in declare block twice, is amenable to loop invariant motion and can be shared.
enum_io3	Instantiate enumeration_io in declare block twice, is amenable to loop invariant motion and can't be shared.
enum_io4	Instantiate float_io in declare block. Is amenable to loop invariant motion.
enum_io5	Instantiate float_io twice in declare block. Is amenable to loop invariant motion and can be shared.
enum io6	Instantiate integer io in declare block. Is amenable to loop invariant motion.
enum_io7	Instantiate integer io twice in a declare block. Is amenable to loop invariant motion and can be shared.
enum_io8	elaborate enumeration_io in library packages. Not sharable.
enum_io9	elaborate sharable versions of enumeration_io in library packages

Problem Test Name	Problem Test Description
ew	This is an example drawn from an Electronic Warfare application feasibility study. The design intends to perform I/O operations using a coprocessor with shared memory (the Z80-FIFO). This was modified for testing purposes to "build in" a set a message to exercise the system. Some of the coding style used in this program is rather strange and the system has not been fully debugged. Some procedures referenced uninitialized variables, and have been modified to avoid this usage (or to insure that these procedures are not called with the test data. As a compilation test, it uses both subunits and packages. The original version was not fully debugged, and the test problem constructed from it does not exercise all the intended functions of the application, because it was not appropriate for ACEC team to complete and debug the code. Some of the original code was modified (marked by "- tcl") to permit execution - the original version contained several errors which were worked around.
filter1	Object oriented design approach to implementing a lag filter. This problem declare the filter parameters as formal parameters to a generic definition of a filter. The generic procedures associated with the filter do not pass any explicit parameters. This test problem resets the filter history.

Problem Test Name	Problem Test Description
filter1i	Object oriented design approach to implementing a lag filter. This problem declare the filter parameters as formal parameters to a generic definition of a filter. The generic procedures associated with the filter do not pass any explicit parameters. This test problem resets the filter history - it is an inlined version of filter1.
filter2	Object oriented design approach to implementing a lag filter. This problem calls on a generic instantiation of a template encapsulating INPUT, OUTPUT, HISTORY, and COEFFICIENT. It advances one time step.
filter2i	Object oriented design approach to implementing a lag filter. This test problem is a variation of FILTER2 with pragma inline specified for procedure ADVANCE.
filter3	Object oriented design approach to implementing a lag filter. This problem passes the filter parameters as actuals to a non-generic procedure. This test problem advances one time step.
filter4	Object oriented design approach to implementing a lag filter. This test problem advances one time step with direct "manual insertion" of source code.
firth1	Example of record assignment
firth1x	Version of FIRTH1 performed with component-by-component assignment. A system with reasonable record processing should perform FIRTH1 and FIRTH1X in comparable times.
firth2	Example of record comparison
firth2x	Example of record comparison with standardized boolean. Optimizing compilers should perform FIRTH2 faster than FIRTH2X.

Problem Test Name	Problem Test Description
firth2y	Example record comparison using component-by-component operations. Many systems will call on a runtime library routine to perform a record comparison. An optimizing compiler which performed this comparison using inline code will execute FIRTH2Y and FIRTH2 in comparable times.
firth3	Example of record aggregate assignment with one component being assigned the same value.
firth3x	Component-by-component version of FIRTH3, with redundant component assignment eliminated.
firth4	Use IN operator which has been observed to do poorly on some systems.
firth4x	Version of FIRTH4 using relational operators rather than IN operator.
firth5	Example where tailoring subprogram linkage conventions can provide significant performance advantages. The set of examples can show whether the system uses different linkage conventions where profitable - whether there are profitable alternatives depends on target machine.
firth5v	Example where tailoring subprogram linkage conventions can provide significant performance advantages.
firth5w	Example where tailoring subprogram linkage conventions can provide significant performance advantages. Uses a local procedure named in a pragma INLINE.
firth5x	Example where tailoring subprogram linkage conventions can provide significant performance advantages.
firth5y	Example where tailoring subprogram linkage conventions can provide significant performance advantages.
firth5z	Example where tailoring subprogram linkage conventions can provide significant performance advantages.

Problem Test Name	Problem Test Description
firth6	Example where constant folding and value propagation will have a large payoff. An optimizing compiler could make this comparable to 3 simple assignments.
firth6x	Example where constant folding and value propagation will have a large payoff. Hand optimized version of FIRTH6.
firth7	Example where constant folding and value propagation will have a large payoff.
firth7x	Hand optimized version of FIRTH7 for comparison.
fold1	The FOLD1-4 set of test problems are intended to determine whether a compilation system is performing loop invariant motion and NOT performing constant folding for integers, by comparing the performance of a set of test problems which could be folded both in contexts where loop invariant motion is possible and where it is not. This test problem is "ii:=100;" which has a simple translation independent of folding and/or loop invariant motion.
fold2	This test problem is "ii:=1+1+1...;" which is amenable to both folding and loop invariant motion.
fold3	This test problem is a call on a procedure containing "ii:=100;"
fold4	This test problem calls on a procedure containing "ii:=1+1+1...;" which is amenable to folding but not loop invariant motion.

Problem Test Name	Problem Test Description
fold5	The FOLD5-8 set of test problems are intended to determine whether a compilation system is performing loop invariant motion and NOT performing constant folding for integers, by comparing the performance of a set of test problems which could be folded both in contexts where loop invariant motion is possible and where it is not. This test problem is "xx:=100.0;" which has a simple translation independent of folding and/or loop invariant motion.
fold6	This test problem is "xx:=1.0+1.0+...;" which is amenable to both folding and loop invariant motion.
fold7	This test problem is a call on a procedure containing "xx:=100.0;" which inhibits both folding and loop invariant motion.
fold8	This test problem calls on a procedure containing "xx:=1.0+1.0+...;" which is amenable to folding but not loop invariant motion.
fold_mod	Optimization test. Constructed so that systems which perform folding will do well.
forward_euler1	Adapted from a radar application. Contains 12 trig function calls, half of which are duplicates (same function, same actual parameters). Compare with Forward.Euler2, which saves function results and omits half the function calls.
forward_euler2	Adapted from a radar application. Compare with Forward.Euler1. This version saves function results and omits half the function calls.
funcexcp	To measure the time associated with cleaning up the stack when an exception is raised during nested function calls.
gamm	Classical test, emphasizes 1-D array access, simple 6 digits precision floating point arithmetic.

Problem Test Name	Problem Test Description
gamm2	Classical test, emphasizes 1-D array access, simple extended precision (9 digit) floating point arithmetic
heapify	Classical test, from CFA study, partial sort.
idioms	Test constructed so that a system which does a good job on common machine idioms will do well. This is a property of both compiler and the target machine architecture.
inst1	test of performance of instantiating and using the generic package ENUMERATION_IO.
inst2	test of performance of instantiating and using the generic package ENUMERATION_IO. This problem uses a previously instantiated package and does a PUT to a string.
inst3	This problem copies from array of strings to string, performing similar operation to INST1, INST2, and INST4.
inst4	test of performance of instantiating and using the generic package ENUMERATION_IO. This problem does a string assignment corresponding to the PUT in INST1 AND INST2 from a variable using the 'image attribute.
inst5	This problem uses a CASE statement to select a simple literal string assignment.
int_0	Timing of a simple task. No interrupt is raised in this test. It is used for purposes of comparison.
int_1	Interrupt test. Time simple interrupt. Raise interrupt and test a flag which is set by the handler. The interrupt task is initiated from the main task.

Problem Test Name	Problem Test Description
int_2	Task switching. An interrupt enables a task with a higher priority than the task which was running when the interrupt occurred. After the interrupt has been serviced, the higher priority task, not the one running when the interrupt occurred, will be scheduled.
int_3	Timing of a simple interrupt is complicated by having several tasks on a wait queue.
int_4	There are several runnable tasks eligible at all times. These tasks have a lower priority than the task performing the null timing loop, and should not be executed.
int_5	An exception is raised within the rendezvous of the interrupt entry call. This problem tests for the performance impact of raising exceptions inside the rendezvous.
int_6	Int_6 is identical to the structure of Int_5 without raising the exception. It is used to compare the time required by exception handling.
int_7	Int_7 tests the response time when an interrupt occurs during an interrupt handler. As the interrupt tasks have the same priority, this test will also check whether an interrupt will override an interrupt handler. The LRM is not clear in specifying that interrupt tasks must have priorities. It is permissible for all interrupts to be treated the same and not preempt each other.
int_8	Int_8 tests the response time when an interrupt occurs during an interrupt handler. Int_8 is similar to Int_7 except the second interrupt task has a higher priority than the executing interrupt. This test will determine whether priorities are recognized by the handler.

Problem Test Name	Problem Test Description
int 9	Int 9 tests the response time when an interrupt occurs during an interrupt handler. Int_8 is similar to Int_7 except the second interrupt task has a higher priority than the executing interrupt. This test will determine whether priorities are recognized by the handler.
invar	Optimization test. Constructed so that systems which do a good job of loop invariant motion will do well.
io0	Language feature test. Text_IO, Set_Col on named file.
io1	Language feature test. Text_IO, OPEN/CLOSE.
io2	Language feature test. Text_IO, Open file, Put 1000 80-character lines, Close file. Writing 80,000 bytes will cause most systems to perform several physical I/O operations.
io3	Language feature test. Text IO, Open file, use Get of character to read the 1000 80-character records written in io2, Close the file.
io4	Language feature test. Text_IO, Open file, use Get Line to read the 1000 80-character records written in io2, Close the file.
io5	Language feature test. Text IO, Open file, use Put_Line to write 1000 80-character records, Close the file.
io6	Language feature test. Text_IO, Open file, use Put to write 100 512-bytes records, Close the file. The records contain 1 6-byte count field, and 406 bytes of blanks.
io7	Language feature test. Text_IO, Open file, use Get_Line to read the 100 512-byte records written in io6, Close the file.
io8	Language feature test. Text_IO, access the end of file function.
io9	Language feature test. Text IO, Reset function.

Problem Test Name	Problem Test Description
io10	Language feature test. Text IO, Is_Open function.
io11	Language feature test, direct file io, Set Index function.
io12	Language feature test, direct file io, Set_Index function followed by a READ. This will read from the same block so no physical IO operations are required. This block is read once outside the loop to set the buffers.
io13	Language feature test, direct file io, Set_Index function, followed by a WRITE. This will write to the same block each time.
io14	Language feature test, direct file io, OPEN/CLOSE.
io15	Language feature test, direct file io. Call on the Index function.
io16	Language feature test. Direct file io; call on Size function.
io17	Language feature test, sequential file, OPEN/CLOSE.
io18	Language feature test, sequential file, OPEN file, WRITE 1000 80 byte records, CLOSE file.
io19	Language feature test, sequential file, OPEN file, READ 1000 80 byte records, CLOSE file This reads the file written by io18.
io20	Language feature test, sequential file, call on END OF FILE function.
io21	Language feature test, sequential file, OPEN file, WRITE 100 511 byte records, CLOSE file.
io22	Language feature test, sequential file, OPEN file, READ 100 80 byte records, CLOSE file. This reads the file written in io21.
io23	Language feature test, sequential file, call on SET_INPUT procedure.
io24	Test of PUT to an interactive console. This problem PUTs an ASCII carriage return

Problem Test Name	Problem Test Description
io25	Test of PUT to an interactive console. This problem PUTs a string and then an ASCII carriage return
io26	Test of PUT to an interactive console. This problem PUTs a string and then an ASCII carriage return
io27	Test of PUT to an interactive console. This problem PUTs an ASCII nul character
io28	put(" a" & ascii.cr); to console. Rewriting the same line to the console repetitively can be optimized by a smart screen manager which could not send any terminal commands which will not change the display. Relative to IO29, this problem performs only one call on procedure PUT.
io29	Put "A" & ascii.cr to console. Rewriting the same line to the console repetitively can be optimized by a smart screen manager which could not send any terminal commands which will not change the display.
io30	Display variable character & ascii.cr. Compare with IO28
io 80 20 1	I/O pattern test problem. The IO 80 20 family of test problems considers random file processing. This problem performs 1_000 reads from direct file with 100 records using a 80-20 distribution. Problem will do well on systems with buffering or cache since the file could fit entirely in memory.
io 80 20 2	I/O pattern test problem. Performs 1_000 reads from direct file with 1_000 records using a 80-20 distribution. Problem will do well on systems with buffering or cache most of time file could fit in memory.
io 80 20 3	I/O pattern test problem. Performs 1 000 reads from direct file with 10_000 records using a 80-20 distribution.

Problem Test Name	Problem Test Description
io_80_20_4	I/O pattern test problem. Performs 1.000 reads from direct file with 10.000 records using a 80-20 distribution, after sorting the requests. Because requests are sorted, the sequence of requests will be strictly non-decreasing and average distance between requests will be small and all the requests to the same block will be processed before another block is accessed. Comparison with io_80_20_3 will show performance effect of sorting batches of transactions.
io_80_20_5	I/O pattern test problem. Performs 1.000 reads from direct file with 100.000 records.
io_80_20_6	I/O pattern test problem. Performs 1.000 reads from direct file with 100.000 records, after being sorted.
io_80_20_7	I/O pattern test problem. Performs 1.000 sequential reads from direct file, resetting after scanning 10 Mbytes. Because the file is large, the value of a <i>small cache is limited</i> . Measurements should reflect physical I/O time.
io_80_20_8	I/O pattern test problem. Performs 1.000 sequential reads from direct file, resetting after scanning 1 Mbytes. Because the file is large, the value of a <i>small cache is limited</i> . Measurements should reflect physical I/O time.
io_80_20_9	I/O pattern test problem. Performs 1.000 sequential reads from direct file, resetting after scanning 10 Kbytes. Because the file is small, the value of a <i>small cache is enhanced</i> .
io_80_20_10	I/O pattern test problem. Performs 1.000 sequential reads from direct file, resetting after scanning 1.000 bytes. Because the file is small, the value of a <i>small cache is enhanced</i> .

Problem Test Name	Problem Test Description
io copy1	I/O pattern test. The IO COPY* problems are tests of sequential processing. This problem copies a sequential file with 500 100-byte records by reading one record at a time and writing it. This produces 1000 I/O operations.
io copy2	I/O pattern test. copy sequential file with 500 100-byte records by reading 100 records into a buffer and writing the buffer. This produces 1000 I/O operations. Could be faster than IO_COPY1 if the system doesn't perform any buffering.
io copy3	I/O pattern test. Copy direct file with 500 100-byte records by reading one records into a buffer and writing the buffer. This produces 1000 I/O operations. Compare with sequential file copy IO_COPY1.
io copy4	I/O pattern test. Copy direct file with 500 100-byte records by reading 100 records into a buffer and writing the buffer. This produces 1000 I/O operations. Could be faster than IO_COPY3 if the system doesn't perform any buffering.
io inter1	I/O pattern test. The IO INTER* family of test problems considers interleaved sequence of simple I/O patterns. This test problem alternately reads from two sequential files, each of 500 100-byte records.
io inter2	I/O pattern test. This test problem alternately reads from a sequential and a direct file of 500 100-byte records.
io inter3	I/O pattern test. This test problem alternately reads from a sequential file and then reads and writes to one record in a direct file.

Problem Test Name	Problem Test Description
io_mem1	I/O pattern test. The IO MEM* family of test problems performs patterns of accessing on an array in memory which can be compared to the corresponding file I/O patterns. This test problem references 1000 records from an array in ascending order. It corresponds to IO_SCAN1.
io_mem2	I/O pattern test. This test problem references 1000 records from an array in descending order. It corresponds to IO_SCAN3.
io_mem3	I/O pattern test. This test problem references 1000 records from an array in descending order. It corresponds to IO_SCAN4.
io_pattern1	I/O pattern test. The IO_PATTERN* family of tests perform simple cyclic patterns of access on a direct file. This problem uses the constant pattern to read 1,000 records - it reads the same record every time. A system with buffering will perform well on this.
io_pattern2	I/O pattern test. This problem uses a two record cycle to read 1,000 records: (r1,r2), (r1,r2), (r1,r2)... A system with buffering will perform well on this.
io_pattern3	I/O pattern test. This problem uses a five record cycle to read 1,000 records: (r1,r2,r3,r4,r5), (r1,r2,r3,r3,r5),... A system with buffering will perform well on this.
io_pattern4	I/O pattern test. This problem uses a ten record cycle to read 1,000 records: (r1,r2,r3,r4,r5,r6,r7,r8,r9,10), (r1,r2,r3,r4,r5,r6,r7,r8,r9,10), ... A system with ten buffers can perform well on this.
io_pattern5	I/O pattern test. This problem uses a one record cycle to write 1,000 records: r1, r1, r1, r1, r1, ... A system with buffers can perform well on this.

Problem Test Name	Problem Test Description
io_pattern6	I/O pattern test. This problem uses a two record cycle to write 1 000 records: (r1,r2), (r1,r2), (r1,r2), (r1,r2), (r1,r2), ... A system with buffers can perform well on this.
io_pattern7	I/O pattern test. This problem uses a five record cycle to write 1 000 records: (r1,r2,r3,r4,r5), (r1,r2,r3,r4,r5), ... A system with buffers can perform well on this.
io_pattern8	I/O pattern test. This problem uses a ten record cycle to write 1 000 records: (r1,r2,r3,r4,r5,r6,r7,r8,r9,r10), (r1,r2,r3,r4,r5,r6,r7,r8,r9,r10), ... A system with ten buffers can perform well on this.
io_recur1	I/O pattern test. The IO-RECUR* family of test problems are composed of cycles of records drawn from a sequence of uniform distributions. This problem selects from the sequence (p1-p10, p11-p110, p111-p10000), ... where the notation pn-pm implies a uniformly distributed record between N and M. This problem models the many disc based tree structures.
io_recur2	I/O pattern test. This problem selects from the sequence (p1-p10, p11-p110), ... This problem models that of many disc based tree structures when the root is forced to be memory resident. Buffers and or caches can aid performance.
io_recur3	I/O pattern test. This problem selects from the sequence (p1-p1, p1-p100) ... (always reads p1) This problem models that of many disc based tree structures when the root is forced to be memory resident. Buffers and or caches can aid performance.

Problem Test Name	Problem Test Description
io_scan1	I/O test pattern. The IO SCAN* family of test problems are simple patterns of access (ascending, descending, uniformly random) which are applied to direct files. This test problem reads 1000 records in ascending order.
io_scan2	I/O test pattern. This test problem reads 1000 records in ascending order with a one millisecond delay between reads. The delay can add enough extra time to force additional disc revolutions between block reads.
io_scan2x	I/O test pattern. This test problem reads 1000 records in ascending order with a ten millisecond delay between reads. The delay can add enough extra time to force additional disc revolutions between block reads.
io_scan3	I/O test pattern. This test problem reads 1000 records in descending order.
io_scan4	I/O test pattern. This test problem reads 1000 records using a random permutation of the records - every record is read.
io_scan5	I/O test pattern. This test problem writes 1000 records to a direct file in ascending order.
io_scan6	I/O test pattern. This test problem writes 1000 records to a direct file in ascending order with a 1 millisecond delay between each write.
io_scan7	I/O test pattern. This test problem writes 1000 records to a direct file in descending order.
io_scan8	I/O test pattern. This test problem writes 1000 records to a direct file using random permutation.
io_scan11	I/O test pattern. This test problem writes 1000 records to a sequential file. Record type is unconstrained string, alternating between 50 and 150 bytes.
io_scan12	I/O test pattern. This test problem reads 1000 records from a sequential file. Record type is unconstrained string, alternating between 50 and 150 bytes.

Problem Test Name	Problem Test Description
io_scan13	I/O test pattern. This test problem writes 1000 variant records (with maximum size 100 characters) in ascending order to a direct file. Alternate between 100 and 7 character records.
io_scan14	I/O test pattern. This test problem reads 1000 variant records in ascending order from a direct file with maximum size of 100 characters. Alternate between 100 and 7 character records.
io_scan15	I/O test pattern. This test problem reads 1000 variant records in random order from a direct file with maximum size of 100 characters. Alternate between 100 and 7 character records.
io_scan16	I/O test pattern. This test problem writes 1000 variant records in ascending order to a direct file with maximum size 100 characters. Alternate between 100 and 7 character records.
io_scan17	I/O test pattern. This test problem reads 1000 variant records in ascending order from a direct file with maximum size 100 characters. Alternate between 100 and 7 character records.
io_scan18	I/O test pattern. This test problem reads 1000 variant records in random order from a direct file with maximum size 100 characters. Alternate between 100 and 7 character records.
io_unif1	I/O pattern test problem. Performs 1,000 reads from direct file with 100 records using a uniform distribution. Systems with buffers or caches may be able to keep entire file in memory.
io_unif2	I/O pattern test problem. Performs 1,000 reads from direct file with 1,000 records using a uniform distribution.

Problem Test Name	Problem Test Description
io.unif3	I/O pattern test problem. Performs 1 000 reads from direct file with 10 000 records using a uniform distribution.
io.unif4	I/O pattern test problem. Performs 1 000 reads from direct file with 10 000 records using a uniform distribution after sorting the records.
io.unif5	I/O pattern test problem. Performs 1 000 reads from direct file with 100 000 records using uniform distribution.
io.unif6	I/O pattern test problem. Performs 1 000 reads from direct file with 100 000 records using sorted uniform distribution.
iqsort	Classical test, variant of quicksort on integers.
kalman	Application study: Kalman Filter For the second release, several uninitialized variables have been assigned values which may modify timings relative to the first release. This program contain dummy routines for I/O operations and may not represent optimum coding of a Kalman filter. However, even if the calculates associated with the filter computations are not correct, as a test problem to exercise an Ada compilation system it will be useful for the ACEC.
kernel1	Classical test, livermore loops, Hydro fragment.
kernel2	Classical test, livermore loops, Incomplete Cholesky - Conjugate Gradient.
kernel3	Classical test, livermore loops, inner product.
kernel4	Classical test, livermore loops, banded linear equations.
kernel5	Classical test, livermore loops, tri-diagonal elimination, below diagonal.
kernel6	Classical test, livermore loops, general recurrence equation.

Problem Test Name	Problem Test Description
kernel7	Classical test, livermore loops, equation of state fragment.
kernel8	Classical test, livermore loops, A.D.I. (Alternate Directions Implicit) integration.
kernel9	Classical test, livermore loops, Integrate predictors.
kernel10	Classical test, livermore loops, difference predictors
kernel11	Classical test, livermore loops, first sum.
kernel12	Classical test, livermore loops, first diff.
kernel13	Classical test, livermore loops, 2-D particle-in-cell (PIC).
kernel14	Classical test, livermore loops, 1-D Particle-in-Cell (PIC).
kernel15	Classical test, livermore loops. Casual FORTRAN development version (recoded in Ada).
kernel16	Classical test, livermore loops, Monte Carlo search loop.
kernel16 goto	Classical test, livermore loops, Monte Carlo search loop; GOTO version.
kernel17	Classical test, livermore loops, implicit conditional computation.
kernel18	Classical test, livermore loops, 2-D explicit hydrodynamic.
kernel19	Classical test, livermore loops, general linear recurrence equations.
kernel20	Classical test, livermore loops; Discrete ordinates transport, conditional recurrence on xx.
kernel21	Classical test, livermore loops, matrix * matrix product.
kernel22	Classical test, livermore loops-Planckian distribution
kernel23	Classical test, livermore loops, 2-D implicit hydrodynamics fragment.
kernel24	Classical test, livermore loops; Find location of first minimum in array.

Problem Test Name	Problem Test Description
label	Observe the performance of a sequence of label null statements.
loop0	Classical test, Knuth loop, find max abs of array.
loop1	Classical test, Knuth loop, multiple matrix by scalar.
loop2	Classical test, Knuth loop, serial search.
loop3	Classical test, Knuth loop, array computations.
loop4a	Classical test, Knuth loop, initialize array with call on pseudo-random number generator. Function is in separate package.
loop4b	Classical test, Knuth loop, initialize array with calls on pseudo-random number generator. Function is declared inline in same unit.
loop4c	Classical test, Knuth loop; Test written as optimized inline code, compare to loop4b.
loop5	Classical test, Knuth loop, exponentials, array references.
loop6	Classical test, Knuth loops, inner loop containing procedure calls.
loop7	Classical test, Knuth loop, squares, sqrt function.
loop8	Classical test, Knuth loop, complex number processing.
loop9	Classical test, Knuth loop, array manipulation.
loop10	Classical test, Knuth loop, conditional testing.
loop11	Classical test, Knuth loop, from FFT.
loop12	Classical test, Knuth loop, 3-D array processing, fairly large basic block.
loop13	Classical test, Knuth loop, binary search.
loop14	Classical test, Knuth loop, arithmetic example.
loop15	Classical test, Knuth loop, 2-D array processing.
loop16	Classical test, Knuth loop, statistical processing call on "erf" function.
loop17	Classical test, Knuth loop, 1 and 2-D array processing
lu	Classical test, LU decomposition (lower-upper matrix decomposition), from CFA study.

Problem Test Name	Problem Test Description
merge1	Classical test, mergesort program run on an unsorted array.
merge2	Classical test, mergesort program run on sorted array
neural	Implementation of an Artificial Intelligence programming technique.
pure1	This test problem could, if Ada functions were required to be "pure" (without side effects and always returning the same function result when given the same inputs) be optimized by folding. Ada functions are not required to be pure, and this problem tests that they have not been improperly optimized.
pure2	This test problem is a version of pure1 which has been hand optimized as if it were pure.
pure3	This test problem could, if Ada functions were required to be "pure" (without side effects and always returning the same function result when given the same inputs) be optimized by loop invariant motion. Ada functions are not required to be pure, and this problem tests that they have not been improperly optimized.
pure4	This test problem is a hand optimized version of PURE3 coded as if function were "pure."
pure5	This test problem is a version of PURE1 using a function which is "pure."
pure6	This test problem is a version of pure5 which has been hand optimized. This is a pure version of pure2

Problem Test Name	Problem Test Description
pure7	This test problem is a version of pure3 which uses a pure function.
pure8	This test problem is a version of pure4 using a pure function.
puzzle	Classical test. F. Baskett's cube placing problem solver.
qsort1	Classical test; median-of-3 nonrecursive quicksort on an unsorted array.
qsort2	Classical test; median-of-3 nonrecursive quicksort on a sorted array.
queens mod	Classical test; Eight queens problem.
reclaim_collection_constrained	Check for reuse of reclaimed space when an ACCESS type to a constrained object type is allocated and then deallocated in a collection. Determine whether space is always, never, or sometimes immediately reused.
reclaim_collection_unconstrained	Check for reuse of reclaimed space when an ACCESS type to a constrained object type is allocated and then deallocated in a collection. Determine whether space is always, never, or sometimes immediately reused.
reclaim_global_heap_constrained	Check for reuse of reclaimed space when an ACCESS type to a constrained object type is allocated and then deallocated in global heap. Determine whether space is always, never, or sometimes immediately reused.
reclaim_global_heap_unconstrained	Check for reuse of reclaimed space when an ACCESS type to an unconstrained object type is allocated and then deallocated in global heap. Determine whether the space is always, never, or sometimes immediately reused.
reed_solomon_0	Error Correcting Code example of bit manipulation. This problem checks that the decoding of an encoded message is the same as the original message.
reed_solomon_1	Error Correcting Code example of bit manipulation. This problem encodes a message.

Problem Test Name	Problem Test Description
reed_solomon_2	Error Correcting Code example of bit manipulation. This problem decodes an error free codeword
reed_solomon_3	Error Correcting Code example of bit manipulation. This problem decodes a codeword which requires correction.
reed_solomon_4	Error Correcting Code example of bit manipulation. This problem decodes an uncorrectable codeword, and raises an exception to indicate this.
runge	Classical test, from CFA study. This problem is one step of a Runge-Kutta iteration. Runge-Kutta is a method of solving differential equations.
s_library_1	This set of test problems measure package elaboration times, in particular library package elaboration time for packages which declare objects of non-static sizes. This problem contains a declare block defining a type, no objects, and a null body. It can be translated into a null.
s_library_2	This is one of the s_library_* family of test problems. This problem contains a declare block defining a type, no objects, and a body with one procedure call.
s_library_3	This is one of the s_library_* family of test problems. This problem contains nested package declarations defining 4 fixed sized arrays. Package body contains one procedure call.

Problem Test Name	Problem Test Description
s.library_5	This is one of the s.library_* family of test problems. A declare block which allocates 4 fixed sized arrays on a heap in a named collection and deallocates them via UNCHECKED_DEALLOCATION. The allocation is expected to be roughly comparable to the overheads of allocating fixed sized objects in a library package.
s.library_6	This is one of the s.library_* family of test problems. A declare block which allocates 4 fixed sized arrays on a heap in a named collection and reclaims space by exiting the block the collection is declared in. The allocation is expected to be roughly comparable to the overheads of allocating fixed sized objects in a library package.
s.library_7	This is one of the s.library_* family of test problems. Measurement of time to elaborate 5 library packages which allocate 4 fixed sized arrays (same 4 as used in the other problems in this set). The measurement of this problem has coarser error bounds because it must use a variation of the timing loop code since it is not possible to force the system to elaborate a library package more than once per program execution.

Problem Test Name	Problem Test Description
s.library_8	This is one of the s.library_* family of test problems. This problem is a versions of s.library_7 using nested packages rather than library packages. It can be much faster than s.library_7 because a simpler (stack based) storage allocation scheme can be used for nested packages.
search	Classical test. Search for a substring in a string.
shell1	Classical test. Shell sort of an unsorted array.
shell2	Classical test. Shell sort of a sorted array.
sieve	Classical test. Determine prime number via Sieve of Erastophanes.
simulate.bmbat	Example extracted from application study
simulate.emrpm	Example extracted from application study
simulate.hmproto	Example extracted from application study
simulate.qmpitch	Example extracted from application study
simulate.rcwfrdet	Example extracted from application study
simulate.umnav	Example extracted from application study
simulate.kmdump	Example extracted from application study
simulate.rmkeying	Example extracted from application study
slice1	Determine efficiency of code for length checks and overlaps for slice assignments.
slice2	Determine efficiency of code for length checks and overlaps for slice assignments.
slice3	Determine efficiency of code for length checks and overlaps for slice assignments.
slice4	Determine efficiency of code for length checks and overlaps for slice assignments.
slice5	Determine efficiency of code for length checks and overlaps for slice assignments.

Problem Test Name	Problem Test Description
slice6	Determine efficiency of code for length checks and overlaps for slice assignments.
slice7	Determine efficiency of code for length checks and overlaps for slice assignments.
slice8	Assign component by component to a slice with literals so there is no possibility of overlap.
ss0	Language feature test, null statement
ss1	Assign floating point variable from literal value.
ss2	Type conversion in static expression – real(1).
ss2_mod1	Type conversion in static expression – real(1).
ss2_mod2	Type conversion in static expression – real(1).
ss3	Assignment of two floating point variables, library scope.
ss4	Floating point addition.
ss5	Floating point multiplication.
ss6	Floating point division.
ss7	Integer literal assignment, literal "1" to library scope variable.
ss8	Type conversion from floating point literal to integer
ss8_mod	Type conversion from floating point literal to integer
ss9	Integer addition.
ss10	Integer division.
ss11	Library scope integer assignment.
ss12	Integer to float type conversion.
ss13	Float to integer type conversion of scalar variable (not a literal as in ss8).
ss14	Test of power function using exp and log function.
ss15	Language feature test, (float) ** 2 which can be treated as (float) * (float).
ss16	Language feature test, (float) ** 3 which can be treated as (float) * (float) * (float).
ss17	Language feature test, assignment to one dimensional array of real.

Problem Test Name	Problem Test Description
ss18	Language feature test, assignment to two dimensional array of real.
ss19	Language feature test, assignment to three dimensional array of real.
ss20	Assignment of library scope floating point variable to local variable.
ss21	Assign float to component of array of records.
ss22	Allocation overhead test. Enter a block containing a statically bounded one dimensional array of float, assign to component of it, and access component to ensure liveness.
ss23	Allocation overhead test. Enter a block containing a statically bounded two dimensional array of float, assign to component of it, and access component to ensure liveness.
ss24	Allocation overhead test. Enter a block containing a statically bounded three dimensional array of float, assign to component of it, and access component to ensure liveness.
ss25	Allocation overhead test. Enter a block containing a dynamically bounded one dimensional array of float, assign to component of it, and access component to ensure liveness.
ss26	Language feature test, GOTO.
ss27	Test of SIN function in math library.
ss28	Test of COS function in math library.
ss29	Language feature test, floating point "abs".
ss30	Language feature test, integer "abs".
ss31	Test of EXP function in math library.
ss32	Test of LOG function in math library.
ss33	Test of SQRT function in math library.
ss34	Test of ARCTAN function in math library.

Problem Test Name	Problem Test Description
ss35	Test of SGN function (on floating point variables) in GLOBAL.
ss36	Language feature test, simple procedure with no parameters; call to library scope procedure – body is null.
ss37	Language feature test, simple procedure with one IN OUT floating point parameters, declared in external library unit – body is null.
ss38	Language feature test, simple procedure with two IN OUT floating point parameters, declared in external library unit – body is null.
ss39	Language feature test, simple procedure with three IN OUT floating point parameters, declared in external library unit – body is null.
ss40	Language feature test, integer unary minus.
ss41	Optimization test for folding of static integer expression, "1+1".
ss41_mod	Optimization test for folding of static integer expression, "1+1". Perform statement in an external procedure to inhibit loop invariant optimization
ss42	Optimization test for folding of static integer expression, "-1".
ss42_mod	Optimization test for folding of static integer expression, "-1". Perform statement in an external procedure to inhibit loop invariant optimization
ss43	Store zero, call procedure, increment integer.
ss44	Optimization test for algebraic simplification "+0" is redundant.
ss45	Assign external integer to zero.
ss46	Assign external integer to "large" literal.
ss47	Optimization test for algebraic simplification, "*1".
ss48	Optimization test for algebraic simplification, "/1".
ss49	Optimization test for algebraic simplification, "*0".

Problem Test Name	Problem Test Description
ss50	Optimization test for algebraic simplification, "***0".
ss51	Optimization test. Check for algebraic simplification, "***1".
ss52	Test use of "INC" instruction for "+1".
ss53	Reference to subscripted array of int, no checking.
ss54	Reference to subscripted array of int, no checking
ss55	Reference array with a constant subscript .
ss56	Optimization test for dead assignment elimination on integers.
ss57	Test subscript computation using FOR loop index.
ss58	Test expression using FOR loop index.
ss59	Unary minus, floating point.
ss60	Assign negative floating literal to scalar.
ss61	Optimization test for algebraic simplification of "* 1.0".
ss62	Optimization test for algebraic simplification of "/ 1.0".
ss63	Optimization test for algebraic simplification of "* 0.0".
ss64	Optimization test for algebraic simplification of "+ 0.0".
ss65	Optimization test for algebraic simplification of "(float) ** 0".
ss66	Optimization test for algebraic simplification of "(float) ** 1".
ss67	Optimization test: algebraic simplification; symbolic simplification of variable divided by itself.
ss68	Optimization test: dead assignment elimination; floating point variable
ss69	Test that parentheses are respected. This is a folded version ss70.
ss70	Test that parentheses are respected. This might be improperly folded into ss69.

Problem Test Name	Problem Test Description
ss71	Optimization test. Assign float variable to itself.
ss72	Language feature test; boolean operator NOT.
ss73	Optimization test: algebraic simplification; boolean NOT NOT.
ss74	Optimization test: algebraic simplification; boolean expressions "AND true" and "OR false".
ss75	Optimization test for common subexpression elimination; array element is referenced twice in same expression.
ss76	common subexpression elimination 9 references to 3D array index computation with different literal terms
ss77	array assignment, coding style aggregate with range specification
ss78	array assignment, coding style aggregate with all elements positionally specified
ss79	array assignment, coding style copy array
ss80	Coding style test. Array assignment using a FOR loop to set each element of 10 component real array to 1.0
ss81	Coding style test. Array assignment using a FOR loop to set the "ith" element of a 10 component integer array to "i".
ss82	if statement, integer relation (true)
ss83	if statement, integer relation similar to ss82, using "not(>=)" rather than "<"
ss84	if statement, integer relation (false), no ELSE clause
ss85	if statement, integer relation (true) with ELSE clause
ss86	if statement, integer relation (false), ELSE clause
ss87	if statement, integer and floating relation (true)
ss88	if statement, integer and floating relation (false) connected with "AND"
ss89	if statement, integer and floating relation (false) "AND THEN"
ss90	if statement, integer and floating relation (true) "OR ELSE" connection

Problem Test Name	Problem Test Description
ss91	if statement, IN operator with static bounds ('range)
ss92	if statement, IN operator with dynamic bound
ss93	if with literal condition, "if false ..."
ss94	if statement, simple boolean variable as condition (false), no ELSE clause
ss95 mod	Make references to local scope variables and avoid easy loop invariant optimization. The set of test problems (ss95_mod, ss96_mod, ss97_mod, and ss98_mod) all include the time to setup the environment, which typically will be much larger than the time to make a reference to a variable. However, with precise time measurements, it will be possible to distinguish between systems which use static-links and those which use a display.
ss96 mod	Reference to intermediate scope variable. One level up
ss97 mod	Reference to intermediate scope variable. Two levels up
ss98 mod	Reference to intermediate scope var. Three levels up.
ss99	String literal assignment .
ss100	Assign one component of an array of records to another
ss101	Standardize boolean. Assign relation on integers to boolean variable.
ss102	Language feature test, MOD operator.
ss103	Language feature test, REM operator.
ss104	FOR loop, range null which is not determinable at compile time. Test of FOR loop setup time.
ss105	FOR loop, containing procedure call.
ss106	FOR loop with null body, could be noop.
ss107	Convert one fixed point type with DELTA of 0.001 to another fixed point type with DELTA of 0.01.
ss108	Convert one fixed point type with DELTA of 0.01 to another fixed point type with DELTA of 0.001.
ss109	Fixed point multiplication.

Problem Test Name	Problem Test Description
ss110	Fixed point addition (no fixed point conversion required).
ss111	String slice assignment (static bounds, 2 character slice).
ss112	Dynamic string slice assignment .
ss113	Catenation operator.
ss114	Record assignment.
ss115	Record component by component assignment (all fields)
ss116	Record assignment, aggregate.
ss117	Raise range constraint, process exception.
ss118	Case statement, compact alternative range.
ss119	Case statement, sparse alternative range.
ss120	Coding style test: polynomial evaluation; Coefficients in array for Horner's rule.
ss121	Coding style test: polynomial evaluation; Explicit powers.
ss122	Coding style test: polynomial evaluation; Inline Horner's rule.
ss123	Coding style test: polynomial evaluation; Preconditioned.
ss124	Call local procedure with 3 default parameters, omitting all parameters on call.
ss125	Call local procedure with 3 default parameters, specify all parameters on call.
ss126	Call local procedure with 3 default parameters, specify second parameters (by name) on call.
ss127	Lower level procedure that ss124-ss126 call on.
ss128	PRED and SUCC functions on enumeration types.
ss129	Same computations as in ss128 on integers .
ss130	Take 'POS attribute of enumeration literal.
ss131	Take attributes 'VAL, 'IMAGE, 'POS, 'VALUE of enumeration type.

Problem Test Name	Problem Test Description
ss132	Comparison between enumeration variable and enumeration literal.
ss133	Case with enumerated type (should be dense jump table)
ss134	Language feature test. Floating point put to string, default exponent field (not 0).
ss135	Language feature test. Floating point Get from string. Exponent field is not zero.
ss136	Language feature test. Floating point Put to string exponent. Field is zero.
ss137	Language feature test. Integer Put to string.
ss138	access IN mode scalar parameter
ss139	assign to OUT mode scalar parameter
ss140	reference IN OUT mode scalar parameter
ss141	call on local function
ss142	call on local inline function
ss143	Call function where actual parameter contains another function call.
ss144	example of textual substitution to compare to ss142
ss145	Reference to IN mode array parameter elements. Size of input array is 100 elements.
ss146	Reference to IN mode array parameter elements. Size of input array is 10 elements.
ss147	Reference to IN mode array parameter elements. Actual parameter is dynamic slice which has bounds of 1 .. 1.
ss148	User-defined generic function.
ss149	inline generic procedure on strings
ss150	inline generic procedure on floating point scalar
ss151	provide example to compare with ss149
ss152	discriminant record assignment
ss153	discriminant record assignment, raising constraint error
ss154	access type reference, checking suppressed
ss155	store into allocated object, checking suppressed

Problem Test Name	Problem Test Description
ss156	field assignments to unpacked record
ss157	field assignment to packed record
ss158	record assignment implying reformatting : unpacked-packed
ss159	record assignment implying reformatting : unpacked-packed
ss160	record field assignment, record movement
ss161	record field assignment, record movement
ss162	allocate 100 linked entries from heap, then follow links and deallocate. This may raise <code>storage_error</code> . This test uses accesses to constrained objects.
ss163	allocate 100 linked entries from heap, then immediately deallocate. An optimizing compiler can omit allocation This shouldn't exhaust space. This test uses accesses to constrained objects.
ss164	allocate 100 linked objects in collection and immediately deallocate them. This test uses accesses to constrained objects. An optimizing compiler can omit allocation.
ss165	allocate 100 linked objects in collection and then follow links and deallocate. If this works once, repetitive executions should not risk raising <code>storage_error</code> . This test uses accesses to constrained objects.
ss166	allocate and follow links. No explicit deallocation on. All space in a collection should be freed on block exit. May raise <code>storage_error</code> . This test uses accesses to constrained objects.

Problem Test Name	Problem Test Description
ss167	allocate 100 linked objects in collection and exit block No explicit deallocation, since space in collection should be freed on block exit. Specifies pragma controlled. May raise storage error. This test uses accesses to constrained objects. An optimizing compiler can omit allocation.
ss168	1D array store with subscript range check
ss169	fetch from 1D array with subscript range checking, using constant subscript
ss170	Fetch from and store into 1D array (same index) on both left and right side of assignment statement with subscript range checking enabled. Subscript computation need only be verified once.
ss171	subscript with FOR loop index (in range) compile time range check possible
ss172	common subexpression elimination subscripts, range checking enabled
ss173	constant term in addressing expression, subscript range checking enabled
ss174	3 references to same array in expression, subscripting expression has constant terms with subscript range checking enabled. Bounds checks can be merged.
ss175	Reference to 4 arrays which overlapping static bounds Can merge bounds checking.
ss176	Optimization test. Problem is amenable to boolean variable elimination.
ss177	Optimization test. Problem has had boolean variable elimination performed by hand.
ss178	Problem has tests which be merged.
ss179	Problem has test merged by hand. Compare with ss178.
ss180	Optimization test. Problem has two separate FOR loops which can be fused.

Problem Test Name	Problem Test Description
ss181	Problem has one loop fused by hand. Compare with ss180
ss182	Loop terminates with "EXIT WHEN ... ; END LOOP;" Simple translation will have conditional branch to exit loop followed by unconditional branch to head of loop. Can be improved by jump tracing into one conditional branch (with reverse condition).
ss183	Loop terminates with "IF ... WHEN EXIT; END IF; END LOOP;" Simple translation will have conditional branch to exit loop followed by unconditional branch to head of loop. Can be improved by jump tracing into one conditional branch (with reverse condition).
ss184	Loop starts "LOOP; EXIT WHEN ... ; ... END LOOP;" Simple translation will have conditional branch to exit loop and unconditional branch at end of loop to the head of the loop. Can be improved by jump tracing into one conditional branch (with reverse condition).
ss185	Control folding "WHILE false LOOP ..." can be translated into a null.
ss186	IF statement with same statement in THEN and ELSE clauses.
ss187	IF statement with null in both THEN and ELSE clauses making test unnecessary.
ss188	integer exponent, **2
ss189	Could fold leading unary minus into a literal further on in the expression. Compare with ss190.
ss190	Hand folded version of ss189.
ss191	integer, exponentiation with variable exponent, (-1)**mm LRM Features : 2.4 4.1.1 4.3 4.5.4 4.5.6 4.6 5.2
ss192	Same subscripting expression of left and right side of assignment statement. Checking suppressed.

Problem Test Name	Problem Test Description
ss193	3 references to same array in expression, subscripting expression has constant terms with subscript range checking suppressed. Subscripting expression has common subexpression.
ss194	Reference to 4 arrays. Compare with ss175. This version suppresses subscript checking.
ss195	superfluous integer assignment
ss196	natural integer multiplication, * 2
ss197	natural divide multiplication, /2
ss198	natural integer multiplication, *4
ss199	natural integer mod, MOD 4
ss200	expression comparable to MOD 4
ss201	natural integer multiplication - not power of 2, *1009
ss202	integer multiplication
ss203	natural division, 1009
ss204	natural integer REM, REM 4
ss205	Subtract two integers and compare result to 0
ss206	Directly compare two integers. Compare with ss205.
ss207	Relational test, compare integer variable against 0
ss208	relational expression, integer / non-zero literal comparison. For comparison with ss207.
ss209	WHILE loop comparable to the FOR loop in ss81
ss210	Expression with common term. Could be optimized.
ss211	Hand optimized common subexpression elimination, uses temporary variable to store common expression.
ss212	example where invariant motion is possible
ss213	example where strength reduction is possible
ss214	machine idiom, reuse of condition code setting. Tests same relation in IF and ELSIF. The first tests for ">" and the second for "<".
ss215	machine idiom, block move? Copy two consecutively allocated fields from one instance of a record type to another. Could be block move here.

Problem Test Name	Problem Test Description
ss216	example floating point, constant folding, constant propagating
ss216.mod	example floating point, constant folding, constant propagating
ss217	example integer point constant folding, constant propagating
ss218	check for invalid algebraic simplification, respect of parenthesis
ss219	foldable real expression. Equivalent to ss216
ss219.mod	foldable real expression. Equivalent to ss216
ss220	algebraic simplification, floating point. Several simplifiable subexpressions
ss221	algebraic simplification, integer. Several simplifiable subexpressions.
ss222	Exponential term in an expression is loop invariant
ss223	relational expression example, OR
ss224	relational expression example "OR ELSE". Same relations as in ss223.
ss225	dead assignments within a loop, killed by assignment after loop exit.
ss226	dead assignments within a block. Variable assigned to local which is not referenced before block is exited.
ss227	example of foldable boolean expression, "OR false"
ss228	example of boolean expression, integer relation OR boolean variable
ss229	example of boolean expression, integer relation OR ELSE boolean variable. Same variables as in ss228
ss230	example of foldable boolean expression, "OR false OR false"
ss231	example of foldable boolean expression "OR ELSE false OR ELSE false"
ss232	example of foldable boolean expression "OR ELSE false"

Problem Test Name	Problem Test Description
ss233	example of type conversion, 2 integer to float conversions
ss234	example of type conversion, integer to float
ss235	This exchanges two elements of a 1D floating point array Same logic as ss150, which is a generic instantiation of an exchange procedure for floating point values.
ss236	example optimizable by application of loop induction
ss237	second example optimizable loop induction
ss238	simple example amenable to loop unrolling
ss239	example of FOR loop with null range, compile time determinable
ss240	simple example amenable to loop unrolling **** How is it different from ss238?
ss241	null type conversion, int() applied to int type variable
ss242	reasonable complex function, composed of pieces presented in ss243 - ss246
ss243	access to array of 2 character strings and assign to a slice
ss244	assign to float field of record
ss245	assignment to discriminant record
ss246	attribute 'POS applied to array element
ss247	function which returns float value
ss248	procedure with OUT mode float parameter
ss249	procedure equivalent to function Max1 in ss141 returns result IN OUT mode parameter
ss250	control flow folding "loop exit; end loop;"
ss251	use VAL, POS, SUCC attributes on enumeration type without representation clauses. This statement enables range checking.

Problem Test Name	Problem Test Description
ss252	use VAL, POS, SUCC attributes on enumeration type without representation clauses. This statement is in a block with suppress RANGE-CHECK. Revision : 11-07-88
ss253	use VAL, POS, SUCC attributes on enumeration type with representation clause and enable range checking
ss254	use VAL, POS, SUCC attributes on enumeration type with representation clauses, suppressing range checking
ss255	uses 'SUCC and 'PRED on enumerated type, enabling range checking
ss256	fetch from access type, pointer to float, enable checking
ss257	store into access to float object, checking enabled
ss258	Pass IN OUT mode array formal parameter, compare with ss259 which calls same procedure but requires type conversion
ss259	Unchecked type conversion on IN OUT mode array formal parameter. Converts an array of real with bounds given by an enumeration type to array of real with bounds given by literal range 1 .. 50.
ss260	local procedure call, body is null
ss261	GOTO next statement. A peephole optimizer should translate this into a null statement.
ss262	example where good register usage would show up. Floating point variable is used in several consecutive IF statements.
ss263	example where good register usage would show up. Variable used in 2 consecutive statements.
ss264	example where good register usage would show up. Integer variable stored in one statement is referenced in relational test and in statement in the THEN clause of the statement.

Problem Test Name	Problem Test Description
ss265	example where good register usage would show up. Integer variable stored in one statement is referenced in the next statement.
ss266	integer abs
ss267	ss267-269 compare the use of a named number, a literal, and an initialized variable to perform some computation. This version uses named number.
ss268	ss267-269 compare the use of a named number, a literal, and an initialized variable to perform the same computation. This version uses a literal number.
ss269	ss267-269 compare the use of a named number, a literal and an initialized variable to perform the same computation. This version uses initialized variable.
ss270	bigint type assignment
ss271	bigint type addition
ss272	bigint type subtraction
ss273	bigint type multiplication
ss274	bigint type division
ss275	bigint type MOD
ss276	bigint type REM
ss277	conversion from int to bigint
ss278	bigint type increment
ss279	bigint type **2
ss280	bigint type relational comparison
ss281	int multiplication
ss282	conversion from bigint to real
ss283	conversion from real to bigint
ss284	fetch from array of bigint. *** How is this different from ss285?
ss285	fetch from array of bigint, fold term into address computation
ss286	extended precision floating point assignment
ss287	extended precision floating point addition

Problem Test Name	Problem Test Description
ss288	extended precision floating point divide
ss289	convert double to real
ss290	convert real to double
ss291	extended precision floating point **2
ss292	extended precision floating point comparison
ss293	extended precision floating point abs
ss294	extended precision floating point sin
ss295	extended precision floating point cos
ss296	extended precision floating point exp
ss297	extended precision floating point LOG
ss298	extended precision floating point sqrt
ss299	extended precision floating point arctan
ss300	convert int to double
ss301	extended precision floating point array assignment
ss302	extended precision floating point literal assignment
ss303	convert integer literal to double
ss304	floating point exponentiation, **16
ss305	floating point exponentiation, **4)**4
ss306	floating point exponentiation, **2)**2)**2)**2
ss307	floating point exponentiation xx:=yy*yy;xx:=xx*xx;xx:=xx*xx;xx:=xx*xx;
ss308	compare with ss304-ss308, xx:= exp(16.0 * log(yy));
ss309	access array of an enumerated type
ss310	assign enumeration literal to variable of type
ss311	explicit raise of user-defined exception, process it
ss312	define user-defined exception, do not raise it
ss313	does not define exception
ss314	test for constant propagation precise floating point literal (9 digits) which can be constant propagated into its following statements and folded.
ss315	hand optimized (folded) version of ss314

Problem Test Name	Problem Test Description
ss316	test for constant propagation *** How is this different from ss317?
ss317	test for constant propagation *** How is this different from ss316?
ss318	use of literal expression in first and second occurrences could be folded
ss319	algebraic simplification, "OR false"
ss320	boolean algebraic simplification, "OR ELSE false"
ss321	boolean algebraic simplification, "OR true"
ss322	boolean algebraic simplification, "OR ELSE true"
ss323	floating point compare against zero
ss324	floating point literal comparison against non-zero
ss325	CASE statement, statically determined
ss326	operations - small unpacked boolean array, =, AND, NOT
ss327	operations on small unpacked boolean array, =, AND
ss328	operations on small unpacked boolean array, /=, AND
ss329	operations on small unpacked boolean array, AND
ss330	operations on small unpacked boolean array, OR
ss331	operations on small unpacked boolean array, OR uses a aggregate with range clause
ss332	operations on small unpacked boolean array, XOR
ss333	operations on small unpacked boolean array, fetch from and store into array element
ss334	operations on small unpacked boolean array, slice assignment
ss335	convert from packed to unpacked small boolean array
ss336	fetch element from small unpacked boolean array
ss337	operations on small packed boolean array, =, AND, NOT
ss338	operations on small packed boolean array, =, AND
ss339	operations on small packed boolean array, /=, AND
ss340	operations on small packed boolean array, AND
ss341	operations on small packed boolean array, OR

Problem Test Name	Problem Test Description
ss342	operations on small packed boolean array, OR uses aggregate with range clause
ss343	operations on small packed boolean array, XOR
ss344	operations on small packed boolean array, fetch from and store into indexed element
ss345	operations on small packed boolean array, slice assignment
ss346	conversion packed to unpacked small boolean array
ss347	fetch element from small packed boolean array
ss348	operations on large packed boolean array, =, AND, NOT
ss349	operations on large packed boolean array, NOT, XOR, AND, OR
ss350	convert unpacked to packed large boolean array
ss351	operations on large unpacked boolean array, =, AND
ss352	operations-large unpacked boolean array, AND, OR, XOR
ss353	convert large packed boolean array to unpacked
ss354	exit from FOR loop with "EXIT WHEN"
ss355	exit from FOR loop with "IF ... THEN EXIT"
ss356	exit from FOR loop with "IF ... THEN GOTO"
ss357	exit from FOR loops nested two deep with "EXIT WHEN"
ss358	call function with default value, specify parameter. Should not evaluate the default expression which is a function.
ss359	call function with default value, omit parameter. Should evaluate the default expression.
ss360	call local procedure
ss361	call null procedure at non-main nesting level
ss362	static expression which must be stored into variable with range constraints with checking enabled
ss363	range checking simplification possible
ss364	range checking, can omit lower limit test since guaranteed by bounds of right hand side
ss365	pass IN OUT mode parameter with range constraint

Problem Test Name	Problem Test Description
ss366	assign literal to variable with range constraints
ss367	assign int variable to int variable with range constraints requiring full upper and lower range checking
ss368	example using properties of built in function to simplify range checking
ss369	force divide by zero and process exception
ss370	call on function returning string
ss371	String slice assignment, lower bound dynamic, upper bound static. Moves a 6 character slice.
ss372	Successive assignment of same variable to two variables with the same range constraints. Can share range checks.
ss373	Assign to an int variable with range constraints. Then assign that value to another variable of the same type. Second assignment need not check constraints again.
ss374	Assign to an int variable with range constraints.
ss375	assign literal to range checked variable
ss376	example with simplifiable control flow
ss377	Loop with procedure call and an unconditional EXIT followed by a statement. Need not generate any code for the statement after the EXIT. Control flow can be simplified.
ss378	call on procedure with IN OUT mode formal parameter (type int)
ss379	make two procedure calls. The lowest level has an exception handler which can (re) raise an exception and propagate it to the next higher level. This problem raises the exception.
ss380	make two procedure calls. The lowest level does not have an exception handler and will simply propagate the exception raised to the next higher level. This problem raises the exception.

Problem Test Name	Problem Test Description
ss381	Block with exception handler which calls on a procedure which raises the exception (the procedure it calls on does not have a handler but simply raises the exception.)
ss382	make two procedure calls. The lowest level has an exception handler which can (re) raise an exception and propagate it to the next higher level. This problem does not raise the exception. In this problem the exception is NOT raised.
ss383	make two procedure calls. The lowest level does not have an exception handler and will simply propagate exception raised to the next higher level. This problem does not raise the exception.
ss384	call on procedure which doesn't propagate exception
ss385	test for loop rotation, WHILE loop. Best code would move the condition test to end of loop and not contain unconditional branch to head of loop.
ss385x	This is GOTO version which has "if ... then goto ..." as end of loop. Should have only one branch. The address from GOTO label should be merged into the relation tests associated with if statement - nonoptimizing compilers might test condition, jump false to "end if", then execute the GOTO, but this is easy to fix with a peephole optimizer. It would be disappointing if the loops constructed by a programmer with "goto" statements are much faster than the "built-in" LOOP constructions, and may encourage poor coding style.
ss386	test for loop rotation. Loop with EXIT WHEN at the beginning of loop. Best code would move the condition test to the end of the loop and conditionally branch back to the head of loop (and insert an initial branch at the initial entry to the loop to go to the test)

Problem Test Name	Problem Test Description
ss387	FOR loop with reverse option This performs the same computations as ss385, ss385x, and ss386, using FOR loop index for counting rather than a global integer variable. If this problem is much faster than the others, the processing of global variables and general arithmetic is suspect.
ss388	sequence of literal assignment statements to array components. Same effect as ss77-ss80
ss389	Do superfluous parentheses produce code? Simple assignment of float variable to another float variable where expression has parentheses.
ss390	Do superfluous parentheses produce code? Add three float variables without any parentheses.
ss391	Do superfluous parentheses produce code? Add 3 float variables, parentheses around first two.
ss392	Do superfluous parentheses produce code? Add 3 float variables, parentheses around last two.
ss393	Do superfluous parentheses produce code? Assign one integer to another with superfluous parentheses around the expression.
ss394	Do superfluous parentheses produce code? Add 3 integer variables, no parentheses.
ss395	Do superfluous parentheses produce code? Add 3 integer variables, parentheses around first two.
ss396	Do superfluous parentheses produce code? Add 3 integer variables, parentheses around last two.
ss397	application function, first order lag filter using floating point variables
ss398	application function, limited integrator using floating point variables
ss399	application function, symmetric deadzone using floating point variables

Problem Test Name	Problem Test Description
ss400	application function, symmetric limiter using floating point variables
ss401	application function, first order lag filter using integers
ss402	application function, limited integrator using integers
ss403	application function, symmetric deadzone, using integers
ss404	application function, symmetric limiter using integers using integers
ss405	nested for loop to access a 2D array
ss406	common subexpression requiring flow analysis
ss407	value loaded in one statement is used in next
ss408	variable and literal are both referenced twice in the same expression
ss409	loop induction example, subscripting with FOR loop index
ss410	search for minimum of array with local function
ss411	search for minimum of array with inline function
ss412	integer variable referenced in one statement is also referenced in next
ss413	order of evaluation test, expression has function call followed by literal term
ss414	order of evaluation test, expression has literal followed by add of a function call. A simple left-to-right order of evaluation would load the literal, save value when it calls on the function, and restore it after the function call.
ss415	order of evaluation test, simple left-to-right order of evaluation will load variable and then have to do a register to register operation to add right hand subexpression. Compare with ss416.

Problem Test Name	Problem Test Description
ss416	order of evaluation test, simple left-to-right order of evaluation of subexpression is almost best here (perform the divide and then add from memory - no need to save and restore the quotient- however, the multiply by 0.5 should be deferred).
ss417	coding style check. assign to variable and depending on results of IF, reassign to it
ss418	coding style check. Depending on IF, assign a value to a variable.
ss419	Pass a scalar parameter which is an element of a dynamically sized array
ss420	Pass a scalar parameter which is an element of a statically sized array
ss421	algebraic simplification, foldable boolean "AND false"
ss422	Strength reduction, by hand, of ss213. Reduces an exponential by FOR loop index "(-1)**i"
ss423	strength reduction test. Has multiply by FOR loop index
ss424	strength reduction test. Hand reduced form of ss423, with multiply by FOR loop index reduced to add.
ss425	strength reduction test. Multiply by induction variable which is not a FOR loop index.
ss426	strength reduction test. Hand reduced form of ss425 with multiply reduced into add. Induction variable is not a FOR loop index.
ss427	data flow. Assign integer to another integer and test if two variables are equal.
ss428	common subscripting expressions, including several indexing expressing across basic block.
ss429	Is constant static array promoted to outer level? uses local constant static array. Compare with ss430
ss430	Is constant static array promoted to outer level? This uses non-local constant array

Problem Test Name	Problem Test Description
ss431	I/O formatting speed and accuracy. Convert powers of two from $2.0^{**}(-75)$ to $2.0^{**}(+75)$ to string variable with PUT and then GET them and compare with original. Remember maximum error and output it after the test. The number of replacements (when a new maximum error is discovered) will vary between systems. However, problem time will be determined mainly by conversion time, and should still be comparable between systems.
ss432	test of algebraic simplification, can be factored
ss433	test algebraic simplification. Factored form of ss432
ss434	test algebraic simplification. Expression with 3 operators. Compare with ss435
ss435	test algebraic simplification. Splits the expression of ss434 into three separate statements with assignments to temporaries.
ss436	test algebraic simplification. Sequence of divisions
ss437	test algebraic simplification. Combine sequence of divisions into multiplications and one division.
ss438	test swapping. FOR loop with embedded IF statement with loop invariant expressions in relation and in the conditional statements. "IF" can be moved out of FOR loop as done by hand in ss439.
ss439	test swapping. Hand optimized version of ss438.
ss440	test of test merging. Several IF's can be merged.
ss441	This version has merged tests, compare with ss440
ss442	register allocation - with call on external procedure, compiler cannot allocate "xx" to register within FOR loop.
ss443	register allocation - no call on "die" so xx can be allocated to register

Problem Test Name	Problem Test Description
ss444	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem enables <code>division_check</code> , <code>overflow_check</code> on floating point division.
ss445	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem enables <code>division_check</code> , <code>overflow_check</code> on int division.
ss446	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem enables <code>division_check</code> , <code>overflow_check</code> on int MOD.
ss447	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem enables <code>division_check</code> , <code>overflow_check</code> on int REM.
ss448	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem suppresses, enables <code>overflow_check</code> on floating point division.
ss449	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem suppresses <code>division_check</code> , enables <code>overflow_check</code> on int division.
ss450	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem enables <code>division_check</code> , suppresses <code>overflow_check</code> on floating point division.
ss451	evaluate effect of selectively suppressing combinations of <code>division_check</code> , <code>overflow_check</code> . This problem enables <code>division_check</code> , suppresses <code>overflow_check</code> on int division.
ss452	language feature test, access the function "clock"

Problem Test Name	Problem Test Description
ss453	language feature test. Call function calendar.seconds
ss454	language feature test. Call real(seconds(clock))
ss455	language feature test. DELAY 0.0 It is permissible for a compilation system to optimize "DELAY 0.0;" into a NULL statement. However, the Ada Uniformity Rapporteur Group (URG) has recommended that implementations consider DELAY 0.0 as a scheduling point. At such a point, an implementation would check whether another task has made it abnormal (i.e. aborted it) and if so would terminate itself. In general it is desirable that all test problems execute as quickly as possible. In this example, the fastest execution time (zero) is not necessarily desirable. On all systems which do not translate this into a NULL the execution time should be as fast as possible.
ss456	language feature test, convert floating point variable to fixed point type DURATION
ss457	language feature test, convert integer to fixed point type duration
ss458	language feature test. Delay 1 millisecond (long enough to actually pause processor). All systems should take 1 millisecond for this statement, plus perhaps some additional scheduling delay, and perhaps some quantization when 1 millisecond is not a model number of type DURATION. This test problem does not follow the general ACEC modeling assumption that execution time will be proportional to a systems factor because all systems should execute for at least the requested delay. This problem is reported with an error code as that MEDIAN will not process statements which do not follow the modeling assumption.
ss459	language feature test. Delay by negative expression
ss460	fixed point arithmetic expression, division by literal

Problem Test Name	Problem Test Description
ss461	fixed point arithmetic, division by variable
ss462	fixed point arithmetic expression, addition
ss463	fixed point arithmetic assignment
ss464	fixed point arithmetic relational test
ss465	fixed point arithmetic expression, subtract literal
ss466	conversion, fixed point to int
ss467	conversion, fixed point to floating point
ss468	conversion, int to fixed point
ss469	reference variable defined in two packages to explore overheads to maintain addressability
ss470	reference variable defined in 1 external packages to explore overheads to maintain addressability
ss471	reference variable defined in local scope plus 3 external packages to explore overheads to maintain addressability.
ss472	reference variable defined in three different packages to explore overheads to maintain addressability
ss473	reference variable defined in four different packages to explore overheads to maintain addressability.
ss474	reference variable defined in three different packages to explore overheads to maintain addressability. Multiple references to packages so might share addressing setup.
ss475	reference variable defined in two different packages to explore overheads to maintain addressability. Reference one package twice.
ss476	reference variable defined in two different packages to explore overheads to maintain addressability.
ss477	reference variable defined in one external package to explore overheads to maintain addressability
ss478	test calling a formal generic procedure. The actual procedure is proc0; an external procedure with a null body.

Problem Test Name	Problem Test Description
ss479	test of coding style variations. Test 'A' through 'Z' for being vowels using a boolean array of character
ss480	test of coding style variations. Test 'A' through 'Z' for being vowels using a local function for is a vowel.
ss481	test of coding style variations. Test 'A' through 'Z' for being vowels using an IF with "OR"s.
ss482	test of coding style variations. Test 'A' through 'Z' for being vowels using a case statement on characters.
ss483	test of coding style variations. Function which accesses a locally declared named number
ss484	test of coding style variations. Function which accesses a outer level named number
ss485	test of coding style variations. Function which accesses a literal.
ss486	test of coding style variations. Set Boolean if character is a vowel using array of Boolean indexed by characters; array has value TRUE for vowels, FALSE for non-vowels.
ss487	test of coding style variations. Set Boolean if character is a vowel using local function.
ss488	test of coding style variations. Set Boolean if character is vowel using case statement on characters.
ss489	test of coding style variations. Set Boolean if character is a vowel using IF and sequence of OR tests
ss490	test of coding style variations. This is NESTED IF version of same condition tested in ss491 using AND THEN.
ss491	test of coding style variations. This is AND THEN version of same condition tested in ss490 using nested IFs.
ss492	test of coding style variations. Set Boolean if character is a vowel using external function.

Problem Test Name	Problem Test Description
ss493	test of coding style variations. Test 'A' through 'Z' for being vowels using an external function.
ss494	test of coding style variations. Simple related to compare with ss495 which will use NOT and inverse relation test.
ss495	test of coding style variations. Simple relation to compare with ss494. This has NOT and inverse relation test.
ss496	test of coding style variations. Can be simplified into ss497 by DeMorgan's rule and eliminate an explicit NOT operator.
ss497	coding style variations. Simplified form of ss496.
ss498	test of coding style variations. Set a boolean to reflect a relation by "if relation then boolean := true ..." Contract with ss499 which does direct assignment of relation into boolean.
ss499	test of coding style variations. Set a boolean to a relation.
ss500	language feature test, unchecked conversion between int and packed array(1..int'size) of Boolean, AND operator on packed Boolean arrays. It should be fairly portable.
ss501	boolean, boolean to integer, AND operator. language feature test, unchecked conversion and boolean operators. Overloaded AND onto integers. Language feature test, unchecked conversions between int and packed array(1..int'size) of Boolean, overloaded AND operator on INT.
ss502	language feature test, unchecked conversion between INT and BOOLARRAY, AND (overloaded on INT as packed Boolean operator), OR (overloaded on INT as packed Boolean operator).
ss503	idioms. Increment and decrement same integer scalar.

Problem Test Name	Problem Test Description
ss504	data flow into noop. Relation test if a variable is not equal to itself.
ss505	data flow into noop. Relation where value of variable is referenced in two subexpressions which are mutually exclusive. No value could satisfy both expressions.
ss506	language feature test, unchecked conversion between INT and BOOLARRAY, AND Boolean array operator.
ss507	machine idioms, test of register reuse. Increment an integer variable, test it in next statement.
ss508	common subexpression shared between the relational test in an IF statement and an arithmetic expression in the THEN clause of the statement.
ss509	common subexpression shared between an arithmetic statement and the relational test in an IF statement that follows it.
ss510	register reuse, test Boolean variable in one statement, next statement is an IF with a relation test using the Boolean variable.
ss511	register allocation. Problem finds the array index of the minimum valued element. This version uses a temporary scalar variable to remember the minimum rather than reference an array element on each FOR loop iteration.
ss512	register allocation. Problem finds the array index of the minimum valued element. This version does not use a temporary scalar variable to remember the minimum but references the array element which has the current minimum on each FOR loop iteration.
ss513	coding style variation, record definition with a default initialization to a float point literal value.
ss514	coding style variation, record definition with a default initialization to a float point scalar variable.

Problem Test Name	Problem Test Description
ss515	coding style variation, record definition with a default initialization, which has an explicit initialization to a floating point literal to override the default. Here the default value is also a literal.
ss516	Coding style variation. For loop index references value of array from prior iteration. Compare with ss517 which saves value from last iteration in temporary scalar.
ss517	Coding style variation. Uses temporary scalar to refer to value of prior iteration. Contrast to ss516
ss518	Sums all elements in a floating point array.
ss519	Problem finds the array index of the maximum valued element. This version uses a temporary scalar variable to remember the minimum rather than reference an array element on each FOR loop iteration. Array is setup in ascending order so each new element is larger than the prior one.
ss520	Problem finds the array index of the maximum valued element. This version doesn't use a scalar variable to remember the maximum but references the array with index of current maximum element on each FOR loop iteration. Array is setup in ascending order so each new element is larger than the prior one.
ss521	call on simple local procedure, no dynamically sized objects, no exceptions, no calls on subprograms.
ss522	Call on simple procedure, similar to that used in ss521, but containing a call on another procedure (which is not executed)
ss523	Call on a (potentially) recursive procedure. Actual control flow is the same as ss521 and ss522. These three problems test for presence of different subprogram linkages for "simple" subprograms.

Problem Test Name	Problem Test Description
ss524	coding style variation. Shift a packed Boolean array using slice assignments. Contrast with ss525 which uses a FOR loop and element by element assignment. Could be implemented as integer divide.
ss525	coding style variation. Shift a packed Boolean array using a FOR loop and element by element assignment. Compare with time for ss524 which uses slice assignment to perform the same actions.
ss526	refer to element of Boolean array with literal subscript.
ss527	conditional raise of user defined exception and go through handler. Taken. Contrast with ss528 where it is not. Explicit RAISE could be implemented by simple branch.
ss528	conditional raise of user defined exception and go through handler. Not taken. Compare with ss527, where exception is raised.
ss529	constant propagating with local variable not visible in any handler
ss530	cse with local variable not visible in handler
ss531	store suppression of local variable visible in handler
ss532	constant propagating : local variable visible in handler
ss533	cse with local variable visible in handler
ss534	store suppression of local variable visible in handler
ss535	sample to embed in code for ss536
ss536	test for loop invariant motion
ss537	named number literal expression is evaluated. Problem tests if expression is evaluated using working precision or rational package.
ss538	Constant real using literal expression is evaluated. Problem tests if expression is evaluated using working precision or rational package. Context is one where the LRM does not require static expression, and working precision is permissible.

Problem Test Name	Problem Test Description
ss539	Literal expression in relational test is evaluated. Problem tests if expression is evaluated using working precision or rational package. Context is one where the LRM does not require static expression, and working precision is permissible.
ss540	Literal floating point expression in assignment statement. LRM does not require evaluation with rational package. Problem also tests precision of evaluation.
ss541	unrolling, test elimination. This has variable upper bound. Could be partially unrolled.
ss542	unrolling, test elimination. This is version of ss541 with literal upper bound. Could be unrolled.
ss542x	Partially unrolled version of ss542 for comparison Eliminates tests for FOR loop index variable=1 within the loop.
ss543	Declare block with null body and exception handler, which is unreachable and superfluous.
ss544	null body check for block overhead
ss545	example where non left to right order of evaluation of one arithmetic expression can enhance performance.
ss546	Call on two parameter function, with left actual parameter a literal and right a further function call. Nested 8 levels. Integer function (max). A strict left to right order of evaluation will result in unnecessary storing and reloading of literal values. Contract with ss547.
ss547	Analogous to ss546 with calls nested on first parameter. A left to right order of evaluation is best here. Good compiler will do both ss546 and ss547 is about the same time.
ss548	Floating point version of ss546. Compare with ss549
ss549	Floating point version of ss547. Compare with ss548.

Problem Test Name	Problem Test Description
ss550	Integer addition with parameters nested on second operand. A left-to-right order of evaluation may generate necessary stores and reloads. Contrast with ss551.
ss551	Integer addition with parameters nested on first operand. A left-to-right order of evaluation is best. Contrast with ss550.
ss552	example where order of evaluation can enhance expression is complex and many targets will run out of registers in evaluating it, provoking register spill
ss553	idioms - common subexpression elimination on subscript on left and right side of an assignment.
ss554	common subexpression elimination for subscripts
ss555	idioms, register reuse. Use INC instruction if available
ss556	Integer constant propagation. Assign zero to a variable, increment variable in next statement.
ss557	register usage. Variable used in indexing expression on left side of assignment statement is also used on right.
ss558	data flow analysis based on values of relations in a conditional statement can fix values of variables in the conditional parts - if they didn't have values which satisfy conditions, code would not execute the alternative, therefore optimized can simplify expressions by using bounds determined by relations.
ss559	comparison for ss558
ss560	algebraic simplification - "-1*"
ss561	comparison to ss560 without explicit multiply
ss561x	data flow analysis based on values of relations in a conditional statement can fix values of variables in the conditional parts - if they didn't have values which satisfy conditions, code would not execute the alternative, therefore optimized can simplify expressions by using bounds determined by relations.

Problem Test Name	Problem Test Description
ss562	language feature test, actual parameter for in out mode scalar includes an expression with function call. Exposed error in implementation in one system which called the function both before and after evaluation.
ss563	inline function with literals, can be folded into assignment of literal value to integer
ss564	inline function test, first actual parameter to max function is 0, permitting simplification.
ss565	inline function test, second actual parameter to max function is 0, permitting simplification.
ss566	reference to first formal integer parameter
ss567	reference to second formal integer parameter
ss568	reference to third formal integer parameter
ss569	reference to fourth formal integer parameter
ss570	reference to fifth formal integer parameter
ss571	reference to sixth formal integer parameter
ss572	reference to seventh formal integer parameter
ss573	reference to eighth formal integer parameter
ss574	reference to ninth formal integer parameter
ss575	reference to first formal float parameter
ss576	reference to second formal float parameter
ss577	reference to third float parameter
ss578	reference to fourth first formal float parameter
ss579	reference to fifth formal float parameter
ss580	reference to sixth formal float parameter
ss581	reference to seventh formal float parameter
ss582	reference to eighth formal float parameter
ss583	reference to ninth formal float parameter
ss584	call procedure with 9 integer parameters
ss585	call procedure with 9 float parameters
ss586	math library test, arcsin
ss587	expression - foldable subexpression using named number
ss588	expression : foldable subexpression using constant real

Problem Test Name	Problem Test Description
ss589	expression with foldable subexpression using variable initialized with literal and not modified
ss590	expression with foldable subexpression using variable initialized with expression and not modified
ss591	comparison with ss587-590, using variable which is modified, but is invariant within the timing loop
ss592	comparison with ss587-591, expression not timing loop invariant
ss593	comparison with ss587-592, variable is global, not invariant
ss594	comparison with ss587 - 593, hand folded version
ss595	expression with foldable subexpression using literals
ss596	call on function returning unconstrained object
ss597	comparison to ss596, omits call on function returning unconstrained object.
ss598	assign to fields of discriminant records, no constraint error will be raised.
ss599	compare with ss598, sequence of assignments to types not in discriminant record
ss600	assign to one field in discriminant records, no constraint error will be raised but will be checked
ss601	assign to variable of same type as field in record, no discriminant checking will be needed. Compare with ss600.
ss602	Assign to field of discriminant records, raise constraint error
ss603	assign whole variant
ss604	reference field in discriminant record, see if system retests discriminant.
ss605	reference field in discriminant record, see if system retests discriminant.

Problem Test Name	Problem Test Description
ss606	Assign literal floating point to scalar in one statement, in next statement assign that scalar to another scalar variable. Could suppress load in second statement.
ss607	Assign literal floating point to scalar in one statement, in next statement assign that scalar to another scalar variable. Could suppress load in second statement. Value is 0.0, so might be able to use some machine idiom to further speed processing.
ss608	Integer version of ss606.
ss609	assign expression to scalar variable in first statement, then assign this variable to another in second statement. Could suppress load in second statement.
ss610	Integer version of ss609.
ss611	Integer version of ss607 or idioms
ss612	Loop which frequently references a variable which might be allocated to a register, or at least loads suppressed
ss613	language feature test, pass parameter to unconstrained formal parameter. With checking enabled. Compare with ss616
ss614	language feature test, pass parameter to unconstrained formal parameter. With checking enabled.
ss615	language feature test, pass parameter to unconstrained formal parameter. With checking enabled.
ss616	language feature test, pass parameter to unconstrained formal parameter. Suppress checking.
ss617	language feature test, pass parameter to unconstrained formal parameter. Suppress checking.
ss618	language feature test, pass parameter to unconstrained formal parameter. Suppress checking.
ss619	6 "GOTO" statements which jump to another "GOTO" statement. Statements are not in order. Test for jump tracing optimization.

Problem Test Name	Problem Test Description
ss620	6 "GOTO" statements which branch to next statement. This is a simpler test for jump tracing than ss619. A peephole optimizer which omits a branch to the next instruction would suffice to optimize this problem.
ss621	language feature test, generic non-inline function, instantiated in external unit
ss622	language feature test, generic inline function, instantiated in external unit
ss623	language feature test, generic non-inline function, instantiated in same unit
ss624	language feature test, generic inline function, instantiated in same unit
ss625	language feature test, local generic inline function,
ss626	language feature test, local generic non-inline function,
ss627	language feature test comparison, non-generic non-inline function,
ss628	language feature test, generic non-inline function, instantiated in external unit
ss629	language feature test, generic inline function, instantiated in external unit
ss630	language feature test, generic non-inline function, instantiated in same unit
ss631	language feature test, generic inline function, instantiated in same unit
ss632	language feature test comparison, non-generic non-inline function,
ss633	language feature test comparison, inline in external package
ss634	sequence of simple assignments. Check for prefetching base addressing overheads, consistency of measurements
ss635	sequence of simple assignments. Check for prefetching base addressing overheads, consistency of measurements

Problem Test Name	Problem Test Description
ss636	sequence of simple assignments. Check for prefetching base addressing overheads, consistency of measurements.
ss637	sequence of simple assignments. Check for prefetching base addressing overheads, consistency of measurements This version has right sides equal to previous statements left sides. Could suppress reload.
ss638	comparison to ss639-check for dead variable elimination.
ss639	dead variable elimination. State never referenced.
ss640	comparison for dead variable elimination. ii global
ss641	comparison for dead variable elimination. No assignments
ss642	Sequence of procedure calls. Timing consistency check.
ss643	floating point cse requiring canonical ordering to detect
ss643x	math library test (float)**(float)
ss644	boolean cse, some canonical ordering applicable tests relation "ll=mm" 5 times, then tests "mm=ll" 5 times
ss645	language feature test, fetch from 1D array
ss646	language feature test, fetch from 2D array
ss647	language feature test, fetch from 3D array
ss648	language feature test, deallocation of null pointer should be "null"
ss649	Follow "if" statement, both legs of which assign to a variable with an assignment to same variable which is independent of the conditional assignments. This makes the assignments within the "if" dead, and the whole "if" statement becomes unnecessary.
ss650	Precede "if" statement, both legs of which assign to a variable with an assignment to same variable which is independent of the conditional assignments. This makes the assignment before the "if" dead.
ss651	Assign to variable within a loop, assignment to be after loop exit, making all assignments within the loop dead.
ss652	Test for packed component spanning a storage unit boundary

Problem Test Name	Problem Test Description
ss653	Test for when unit doesn't cross unit boundary
ss654	Test for both spanning and nonspanning accesses
ss655	This tests packing and unpacking
ss656	Simple assignment
ss657	Test for packed component spanning a storage unit boundary
ss658	Test for when unit doesn't cross unit boundary
ss659	Test for both spanning and nonspanning accesses
ss660	This tests packing and unpacking
ss661	Simple assignment
ss662	Test for packed component spanning a storage unit boundary
ss663	Test for when unit doesn't cross unit boundary
ss664	Test for both spanning and nonspanning accesses
ss665	This tests packing and unpacking
ss666	Simple assignment
ss667	Test for packed component spanning a storage unit boundary
ss668	Test for when unit doesn't cross unit boundary
ss669	Test for both spanning and nonspanning accesses
ss670	This tests packing and unpacking
ss671	Simple assignment
ss672	Test for packed component spanning a storage unit boundary
ss673	Test for when unit doesn't cross unit boundary
ss674	Test for both spanning and nonspanning accesses
ss675	This tests packing and unpacking
ss676	Simple assignment
ss677	Test for packed component spanning a storage unit boundary
ss678	Test for when unit doesn't cross unit boundary
ss679	Test for both spanning and nonspanning accesses
ss680	This tests packing and unpacking

Problem Test Name	Problem Test Description
ss681	Simple assignment
ss682	Test left justified boolean field
ss683	Test the boolean field next to the left justified field
ss684	Test the boolean field next to the right justified field
ss685	Test right justified boolean field
ss686x	Bit manipulation using array indexing. Doing same computations and ss686y.
ss686y	Bit manipulation using array wide logical operators
ss687	Test for packed component spanning a storage unit boundary
ss688	Test for when unit doesn't cross unit boundary
ss689	Test for both spanning and nonspanning accesses
ss690	This tests packing and unpacking
ss691	Simple assignment
ss692	Test for packed component spanning a storage unit boundary
ss693	Test for when unit doesn't cross unit boundary
ss694	Test for both spanning and nonspanning accesses
ss695	This tests packing and unpacking
ss696	Simple assignment
ss697	Test for packed component spanning a storage unit boundary
ss698	Test for when unit doesn't cross unit boundary
ss699	Test for both spanning and nonspanning accesses
ss700	This tests packing and unpacking
ss701	Simple assignment
ss702	Test for packed component spanning a storage unit boundary
ss703	Test for when unit doesn't cross unit boundary
ss704	Test for both spanning and nonspanning accesses
ss705	This tests packing and unpacking
ss706	Simple assignment

Problem Test Name	Problem Test Description
ss707	Test for packed component spanning a storage unit boundary
ss708	Test for when unit doesn't cross unit boundary
ss709	Test for both spanning and nonspanning accesses
ss710	This tests packing and unpacking
ss711	Simple assignment
ss712	Test for packed component spanning a storage unit boundary
ss713	Test for when unit doesn't cross unit boundary
ss714	Test for both spanning and nonspanning accesses
ss715	This tests packing and unpacking
ss716	Simple assignment
ss717	Test left justified boolean field, record has a representation clause
ss718	Test the boolean field next to the left justified field, record has a representation clause
ss719	Test the boolean field next to the right justified field, record has a representation clause
ss720	Test right justified boolean field, record has a representation clause
ss721	Test fixed point assignment with no conversion Dummy version of ss721 which will generate an error message to be processed by FORMAT. Helps automate the test suite.
ss722	Test simple fixed point conversion Dummy version of ss722 which will generate an error message to be processed by FORMAT. Helps automate the test suite.
ss723	Test the fixed point conversion from base 10 to base 2 Revised : 11-07-88 to reflect SPR#16 Dummy version of ss723 which will generate an error message to be processed by FORMAT. Helps automate the test suite.

Problem Test Name	Problem Test Description
ss724 mod	Test field not on storage unit boundary
ss725 mod	Test field on storage unit boundary
ss726 mod	Test creation of user-defined named number. User-defined named numbers should be treated as literals as SS729.MOD
ss727 mod	Test access to system-defined name numbers. Access to system defined named numbers should be treated as literals as in SS729.MOD.
ss728 mod	Test access of user-defined name numbers. User-defined named number should be treated as literals as in SS729 MOD.
ss729 mod	Test access of literal expression. Literal usage should be folded.
ss730 mod	Test ADDRESS attribute of a subroutine.
ss731 mod	Test ADDRESS attribute of a package object. Revision : 11-07-88
ss732 mod	Test ADDRESS attribute of a dynamic object.
ss734 mod	Test SIZE attribute of a statically allocated object.
ss735 mod	Test SIZE attribute of a dynamically allocated object.
ss736 mod	Test POSITION attribute for a record component.
ss737 mod	Test FIRST BIT attribute for a record component.
ss738 mod	Test LAST BIT attribute for a record component.
ss739 mod	Test STORAGE TYPE attribute for an access type.
ss740 mod	Test STORAGE TYPE attribute for a task type.

Problem Test Name	Problem Test Description
ss741	TEST FOR STORAGE RECLAMATION IN COLLECTION ACCESSED WITH UNCONSTRAINED TYPES. Allocate 52 small objects, filling up about 17% of the collection. Then deallocate them, and do it again, deallocating in reverse order. Then allocate 20 large objects filling up the same fraction of the collection. Then deallocate these objects. Repeat all the above 2*rep_count times. If the deallocated space is not reused, later allocations will fail. The collection contains space for approximately rep_count times 1000 "int" variable. Because of possible space overhead for allocated objects, the number of actual entries the collection can contain is permitted by the LRM to vary between implementations. This test problem allows a "fudge factor" of 3. For a tight system, rep_count could be 1, which would fill up collection to a bit over 50% each time, and executing the allocation of large objects would check that the space for small ones had been reused. However, for many heap management algorithms, this would provoke failure, so the problem does not try to pack the collection as tight as might be possible. In any case, the problem will fail, raising STORAGE_ERROR, when deallocated storage is not reused.
ss744	Test using a single shared scalar variable
ss745	Test using a pair of shared scalar variables
ss746	Test using shared access variables
ss747	Test interface to null assembly language routine. Interface to assembler is a Chapter 13 feature not required to be supported on all implementations.
ss748	Access component selected by function call on the left side of assignment statement.

Problem Test Name	Problem Test Description
ss749	Optimization test for invariant loop code motion. This example contains an invariant expression in an inner loop. For the sake of comparison, see tests ss750 and ss222.
ss750	Test for loop interchange optimization.
ss751	Optimization test: omission of an unreachable assignment
ss752	Optimization test: invariant motion; integer assignment statement.
ss753	Assign out of range static expression to an integer with range constraints. See if it optimizes into a simple raise of CONSTRAINT_ERROR.
ss754	Explicit IF statement which tests static expression out of range and raises CONSTRAINT_ERROR.
ss755	Assign out-of-range static expression to a variable with range constraints. Null handler.
ss756	Range checking would be verified at compile time
ss757	Assign out of range dynamic expression. Compare with ss753 through ss756. This needs explicit checking.
ss758	Fetch value from one dimensional array. No constraint checking.
ss759	Fetch value from two dimensional floating point array. No constraint checking.
ss760	Fetch value from three dimensional floating point array. No constraint checking.
ss761	Fetch value from one dimensional array. Constraint checking.
ss762	Fetch value from two dimensional floating point array. Constraint checking.
ss763	Fetch value from three dimensional floating point array. Constraint checking.
ss764	Bit manipulation using array aggregate. Set a component of packed boolean array to TRUE by using an OR against a variable.

Problem Test Name	Problem Test Description
ss765	Bit manipulation using array aggregate. Set a component of packed boolean array to TRUE by using an OR using an aggregate.
ss766	Bit manipulation using indexing. Set element of packed boolean array to true, selected by literal subscript. Similar to ss765.
ss767	Bit manipulation using indexing. Set dynamically computed element of packed boolean array to true. Similar to ss765.
ss768	Bit manipulation using indexing. Set dynamically determined element of packed boolean array to true, using array of bits and OR operator. Similar to ss767.
ss769	Consistency check. same problem as ss36, ss770-773
ss770	Consistency check. same problem as ss36,ss769,ss771-773
ss771	Consistency check. This is same problem as ss36, ss769-770, ss772-ss773
ss772	Consistency check. This is same problem as ss36, ss769-771, ss773
ss773	Consistency check. same problem as ss36, ss769 - ss772
ss774	declare and reference array, setup by copying from another array declared at higher level. Compare this with other ways of setting up an initialized array.
ss775	declare and reference static array, setup with static aggregate. Optimizing compiler could simply make array references directly refer to the static aggregate and not do a copy.
ss776	declare and reference array, setup by executing code in a loop in body of the block.
ss777	reference array setup in higher level. No initialization code need be done here.
ss778	declare and reference array, setup by aggregate with "others" clause

Problem Test Name	Problem Test Description
ss779	Reference the 0th real variable declared in a package. Early variables may be able to use short displacements.
ss780	Reference the 2nd real variable declared in a package. Early variables may be able to use short displacements.
ss781	Reference the 8th real variable declared in a package. Early variables may be able to use short displacements.
ss782	Reference the 16th real variable declared in a package. Early variables may be able to use short displacements.
ss783	Reference the 32th real variable declared in a package. Early variables may be able to use short displacements.
ss784	Reference the 64th real variable declared in a package. Early variables may be able to use short displacements.
ss785	Reference the 128th real variable declared in a package. Early variables may be able to use short displacements.
ss786	Reference the 256th real variable declared in a package. Early variables may be able to use short displacements.
ss787	Reference the 512th real variable declared in a package. Early variables may be able to use short displacements.
ss788	Reference the 1024th real variable declared in a package. Early variables may be able to use short displacements.
ss789	Reference the 2nd real field declared in a record. Early fields may be able to use short displacements.
ss790	Reference the 8th real field declared in a record. Early fields may be able to use short displacements.
ss791	Reference the 16th real field declared in a record. Early fields may be able to use short displacements.
ss792	Reference the 32th real field declared in a record. Early fields may be able to use short displacements.
ss793	Reference the 64th real field declared in a record. Early fields may be able to use short displacements.
ss794	Reference the 128th real field declared in a record. Early fields may be able to use short displacements.

Problem Test Name	Problem Test Description
ss795	Reference the 256th real field declared in a record. Early variable may be able to use short displacements.
ss796	Reference the 512th real field declared in a record. Early fields may be able to use short displacements.
ss797	Reference the 1024th real field declared in a record. Early fields may be able to use short displacements.
ss798	Reference the 0th real field declared in a record. Early fields may be able to use short displacements.
ss799	Addition of variables of type CALENDAR.TIME
ss800	Greater than comparison of type CALENDAR.TIME
ss801	Equal comparison of type CALENDAR.TIME
ss802	Comparison of type DURATION with SECONDS(TIME2)
ss803	Call on CALENDAR.TIME.OF
ss804	Assign an access type to NULL.
ss805	Exchange two non-null access type variables.
ss806	Test of math_dep.INTEXP function on literal
ss807	Test of math_dep.ADX function on literals
ss808	Test of math_dep.SETEXP function on literals
ss809	Test of math_dep.INTEXP function on variables
ss810	Test of math_dep.ADX function on variables
ss811	Test of math_dep.SETEXP function on variables
ssearch	Serial search 1D array of float.
ssearch2	Serial Search 1D array of float for matching component, unrolling the search loop 4 times.
strength	Optimization test. Constructed so that systems which do strength reduction will do well.
tak	Classical test of procedure calling and parameter passing. This is the "TAK" function in Ada adapted from the book "Performance and Evaluation of Lisp Systems," by R. Gabriel, MIT Press, 1985. It is a variant of a program originally developed by Ikuo Takeuchi.
target	classical test from CFA study, target tracking

Problem Test Name	Problem Test Description
task1	language feature test, task creation and termination
task2	language feature test, task creation and termination. Uses a procedure which declares 10 tasks of the same task type. Implies synchronization with these tasks to terminate the procedure.
task3	Simple rendezvous with task making entry call arriving at rendezvous first.
task4	Language feature test of aspects of tasking. Passes a nonscalar parameter. Conditional select statement with WHEN clause.
task5	Language feature test of aspects of tasking. Always takes ELSE alternative.
task6	Language feature test of aspects of tasking. Tests access time to dynamic attributes.
task7	Language feature test of aspects of tasking. Classic test. Uses rendezvous with scalar parameters, simple arithmetic.
task8	Language feature test of aspects of tasking. Classic test. Uses rendezvous with scalar parameters, simple arithmetic.
task9	Language feature test of aspects of tasking. Classic test. Uses rendezvous with scalar parameters, simple arithmetic.
task10	Language feature test of aspects of tasking. Classic test. Uses rendezvous with scalar parameters, simple arithmetic.
task11	Language feature test of aspects of tasking. Entering tasks make a sequence of entry calls. Accepting task contains a sequence of ACCEPTs with DOs containing a simple code sequence. Accepting task will arrive at rendezvous first.

Problem Test Name	Problem Test Description
task12	Language feature test of aspects of tasking. Entering tasks make a sequence of entry calls. Accepting task contains a SELECT with a list of alternatives with a DO containing a simple code sequence. Accepting task will arrive at rendezvous first.
task13	language feature test of aspects of tasking. Entering tasks make a sequence of entry calls. Accepting task contains a sequence of ACCEPTs with DO containing a simple code sequence. Some code outside of the rendezvous. Accepting task will arrive at rendezvous first.
task14	Language feature test of aspects of tasking. Entering tasks make a sequence of entry calls. Accepting task contains a select statement with WHEN clauses, only one of which is satisfied. Accepting task will arrive at rendezvous first.
task15	Language feature test of aspects of tasking. Entering tasks make a sequence of entry calls. Accepting task contains a conditional select statement with WHEN clauses, none of which is satisfied. Accepting task will arrive at rendezvous first.
task16	Language feature test of aspects of tasking. Entering tasks make a sequence of entry calls. Accepting task contains a conditional select statement with WHEN clauses, none of which is satisfied. No open alternative, no waiting tasks. Selected alternative is "DELAY minus one;". Accepting task will arrive at rendezvous first.

Problem Test Name	Problem Test Description
task17	Language feature test of aspects of tasking. Accepting task contains a conditional select statement with WHEN clauses, all of which are satisfied. There are no tasks waiting on accept. The selected alternative will be null.
task18	Language feature test of aspects of tasking. Accepting task contains a conditional select statements with WHEN clauses, none of which is satisfied. Tasks waiting at all entries. Selected alternative is ELSE. This test measures the time to test alternatives – should be fairly fast.
task19	Language feature test of aspects of tasking. Accepting task contains a conditional SELECT statement with WHEN clauses, only one of which is satisfied. Should be tasks waiting on all entries. Accepting task will arrive at rendezvous first. Test measures evaluation of alternative guards and picking one alternative.
task20	Language feature test of aspects of tasking. Entering tasks makes sequence of entry calls. Accepting task contains a SELECT statement with WHEN clauses, only one of which is satisfied. Entering task will arrive at rendezvous first.
task21	Language feature test of aspects of tasking. Entering tasks make a sequence of entry calls. Accepting task contains conditional SELECT with guards, only one of which is satisfied. Entering task will arrive at rendezvous first. Entering task will arrive at rendezvous first.

Problem Test Name	Problem Test Description
task22	Language feature test of aspects of tasking. Entering tasks makes sequence of entry calls. Accepting task contains conditional SELECT with guards, only one of which is satisfied. Accepting task will arrive at rendezvous first.
task23	language feature test of aspects of tasking. Simple rendezvous : task doing accept arrives at rendezvous first
task24	Language feature test of aspects of tasking. Simple rendezvous with task making entry call arriving at rendezvous first. Task performing the accept is in a subunit.
task25	Language feature test of aspects of tasking. Classic test. Uses rendezvous with scalar parameters, simple arithmetic.
task26	language feature test of aspects of tasking. Simple rendezvous with task making entry call arriving at rendezvous first. Task performing the accepts is in separate package.
task27	Language feature test of aspects of tasking. Rendezvous with the task making an entry call arriving at rendezvous first. Brackets each ACCEPT with a SELECT/END pair.
task28	Language feature test of aspects of tasking. Rendezvous with the task making an ENTRY call arriving first. Brackets each ACCEPT with a SELECT/ END pair containing WHEN clauses which can be evaluated at compile time.

Problem Test Name	Problem Test Description
task29	Language feature test of aspects of tasking. Rendezvous with the task making an ENTRY call arriving at the rendezvous first. Brackets each ACCEPT with a SELECT/END pair containing WHEN clauses which can be evaluated at compile time.
task30	Language feature test of aspects of tasking. Selective wait in accepting task. Task will be waiting on accept and DELAY alternative will not be taken. It should not be necessary to setup the DELAY and then cancel it.
task31	Language feature test of aspects of tasking. Selective wait in accepting task. Will take the delay alternative, and then the entry call will be made. With canonical implementation, the system will have to cancel the DELAY notification it sets up to awaken the task if an entry was not made within the specified delay.
task32	Language feature test of aspects of tasking. Conditional wait in accepting task, will always take the ELSE alternative.
task33	Language feature test of aspects of tasking. Conditional wait in accepting task, will always take the ELSE alternative.
task34	Language feature test of aspects of tasking. Selective wait in accepting task. Will always take the delay alternative, which will expire without an entry call being made.
task34_delta	Simple delay to compare with task34.
task35	Language feature test of aspects of tasking. Selective wait in accepting task. Will always take the DELAY alternative, which will expire without an entry call being made.
task35_delta	Provide a delay 0.0 for comparison with task35.

Problem Test Name	Problem Test Description
task36	Language feature test of aspects of tasking. Entering task makes a sequence of calls. Accepting task contains a conditional SELECT with ELSE alternatives which are never taken. Entering task will arrive at rendezvous first.
task37a	Language feature test of aspects of tasking. This program raises a user-defined exception inside a rendezvous.
task37b	Language feature test of aspects of tasking. Compare this program to task37a. Creates and normally terminates the tasks while task37a forces abnormal termination of test.
task38	Language feature test of aspects of tasking. This program makes an entry call on an aborted task. Will raise a TASKING ERROR exception.
task39	Language feature test of aspects of tasking. This program aborts a task which is already aborted.
task40	Language feature test of aspects of tasking. This program creates a task which aborts itself.
task41	Language feature test of aspects of tasking. Simple rendezvous. One task in library unit. Accepting task will arrive at rendezvous first.
task42	Language feature test for aspects of tasking. Simple rendezvous with equal priority tasks, one task in library unit.
task43	Language feature test for aspects of tasking. Simple rendezvous with equal priority tasks, both tasks in same compilation unit.
task44a	Higher priority task becoming eligible to run during rendezvous of lower priority tasks.
task44b	Higher priority task becoming eligible to run during rendezvous of lower priority tasks.

Problem Test Name	Problem Test Description
task45a	Higher priority task becoming eligible to run during running of lower priority task.
task45b	Higher priority task becoming eligible to run during running of lower priority task.
task46	Task with a terminate option which is taken.
task46x	Task with a terminate option which is not taken.
task47	Bounded buffer with scalar parameter
task48	Entries tied to interrupt are called directly.
task49	A call to one of an entry family.
task50	The LRM does not require that there be a unique accept statement for each entry. The purpose of this test is to present a test problem which has multiple accept statements for the same entry to see if this introduces additional overhead.
task51	To dynamically allocate a new record which contains a task of higher priority than the allocating task. The created task does nothing. This is a test of dynamic task creation/termination.
task52	To measure the time for a simple rendezvous with a task created as a component of a dynamically allocated record. The time of task creation/termination is excluded.
task53	To measure the time for a task to abort a different task. The aborted task is also created on each iteration.
task54 mod	To measure the time required when a task specifies an inadequate storage_size with a static expression. The resulting exception is handled with a null statement.
task55_mod	To measure the time required when a task specifies an inadequate storage size with a dynamic expression. The resulting exception is handled with a null statement.

Problem Test Name	Problem Test Description
task56	To measure the time required when a task specifies an adequate STORAGE.SIZE with a static expression. The task body performs a simple call on proc0.
task57	To measure the time for a simple rendezvous with a task created as a component of a dynamically allocated record. The time of task creation/termination is excluded.
task58	To measure the time required when a task has multiple open accepts. The accepts can theoretically be done in a nondeterministic order. Each open select will be accepted one time for each iteration of the timing loop. The test notes whether or not the accepts were done in lexical order.
task59	To evaluate the performance of a task with multiple delay alternatives. The LRM permits a select to have multiple delay alternatives.
task60	To measure the time required when a task has multiple accepts and is forced to process them in lexical order by opening them one at a time in lexical order. This is done in order to compare with the nondeterministic case. Each select will be opened and accepted one time (in lexical order) for each iteration of the timing loop.
task_num_1	Test the performance of rendezvous when you vary the number of tasks. This test has only 1 task.
task_num_5	Test the performance of rendezvous when you vary the number of tasks. This test has 5.
task num 10	Test the performance of rendezvous when you vary the number of tasks. This test has 10.
task num 15	Test the performance of rendezvous when you vary the number of tasks. This test has 15.
task_num_20	Test the performance of rendezvous when you vary the number of tasks. This test has 20.

Problem Test Name	Problem Test Description
task_num_25	Test the performance of rendezvous when you vary the number of tasks. This test has 25.
task_num_30	Test the performance of rendezvous when you vary the number of tasks. This test has 30.
task2_num_1	Test the performance of rendezvous when you vary the number of tasks. This test has 1.
task2_num_5	Test the performance of rendezvous when you vary the number of tasks. This test has 5.
task2_num_10	Test the performance of rendezvous when you vary the number of tasks. This test has 10.
task2_num_15	Test the performance of rendezvous when you vary the number of tasks. This test has 15.
task2_num_20	Test the performance of rendezvous when you vary the number of tasks. This test has 20.
task2_num_25	Test the performance of rendezvous when you vary the number of tasks. This test has 25.
task2_num_30	Test the performance of rendezvous when you vary the number of tasks. This test has 30. This will queue up 31 entry calls on one accept before being processed.
trie1	A trie test, this test inserts each of 20 keys in the trie in ascending order, and then deletes them in descending order. It will fail if a duplicate is found during insertion, or if a key is not found during deletion. A TRIE is also known as a digital search tree. Refer to The Art of Computer Programming, Volume 3, Searching and Sorting, by D. Knuth, Addison Wesley, 1973, for a detailed discussion.

Problem Test Name	Problem Test Description
trie2	A trie test, this problem searches for each key present in trie, and for an equal number of keys which are not present in the trie. It will fail if the records in the trie are not found, or if records which are not in the trie are found. Trie size is 20 records.
unreach	Optimization test. Systems which do a good job of eliminating unreachable code will do well on. Primarily a test of space.
whet1	classical test (whetstone), general workload
whet2	Classical test. Whetstone benchmark with suppression of constraint checking.
whet3	Classical test, Whetstone benchmark, extended precision, checking enabled.
whet4	Classical test. Whetstone benchmark, single precision, optimize=space, checking suppressed.

5.2 Appendix II, TEST PROBLEM TO SOURCE FILE MAP

This appendix provides a cross reference between the test problem name and the file that contains it.

Problem Test Name	Problem Source File Name
a_star	A_STAR
acker1	HANSON
acker2	ACKER2
activation1	ACTIVE
activation2	ACTIVE
ai_create_delete_kb	AIFRAME
ai_create_object	AIFRAME
ai_load_kb_from_file	AIFRAME
ai modify object	AIFRAME
ai query	AIFRAME
alias1	ALIAS
alias2	ALIAS
alias3	ALIAS
alias4	ALIAS
alias5	ALIAS
alias5x	ALIAS
alias6	ALIAS
alias6x	ALIAS
alias7	ALIAS
alias7x	ALIAS
alias8	ALIAS
alias8x	ALIAS
alias9	ALIAS
alias10	ALIAS
alias11	ALIAS
alias12	ALIAS
alias13	ALIAS
alias14	ALIAS
alias15	ALIAS
alias16	ALIAS
arti_asum	ARTI
arti_atan2	ARTI

Problem Test Name	Problem Source File Name
arti.cos	ARTI
arti.fmod	ARTI
arti.ifpm.control	ARTI
arti.ifpm.init	ARTI
arti.ifpm.io	ARTI
arti.ifpm.rotors	ARTI
arti.nairini	ARTI
arti.nscni	ARTI
arti.nutmini	ARTI
arti.sin	ARTI
async1	ASYNC1
async2	ASYNC2
async3	ASYNC3
async4	ASYNC4
async5	ASYNC5
auto	CFA
avl_0	AVL
avl_1	AVL
avl_2	AVL
avl_3	AVL
avl_4	AVL
avl_5	AVL
avl_6	AVL
avl_7	AVL
avl_8	AVL
avl_9	AVL
avl_10	AVL
avl_11	AVL
bmt	CFA
bsort1	SORT
bsort2	SORT
cat1	SLICE
cat2	SLICE

Problem Test Name	Problem Source File Name
cat3	SLICE
cio1	CIO
cio2	CIO
cio3	CIO
cio4	CIO
cio5	CIO
cio6	CIO
cio7	CIO
cio8	CIO
cio9	CIO
cio10	CIO
cio11	CIO
cio12	CIO
cio13	CIO
cio14	CIO
ciqsort	CIQSORT
claim01	CLAIM01
claim02	CLAIM02
claim03	CLAIM03
claim04	CLAIM04
claim05	CLAIM05
claim06	CLAIM06
claim07	CLAIM07
claim08	CLAIM08
claim09	CLAIM09
claim10	CLAIM10
claim11	CLAIM11
claim12	CLAIM12
claim13	CLAIM13
claim14	CLAIM14
claim15	CLAIM15
claim16	CLAIM16
claim17	CLAIM17

Problem Test Name	Problem Source File Name
claim18	CLAIM18
claim19	CLAIM19
claim20	CLAIM20
claim21	CLAIM21
claim22	CLAIM22
claim23	CLAIM23
claim24	CLAIM24
claim25	CLAIM25
claim26	CLAIM26
claim27	CLAIM27
claim28	CLAIM28
claim29	CLAIM29
claim30	CLAIM30
claim31	CLAIM31
claim32	CLAIM32
claim33	CLAIM33
claim34	CLAIM34
claim35	CLAIM35
claim36	CLAIM36
claim37	CLAIM37
claim38	CLAIM38
claim39	CLAIM39
claim40	CLAIM40
claim41	CLAIM41
claim42	CLAIM42
claim43	CLAIM43
claim44	CLAIM44
claim45	CLAIM45
claim46	CLAIM46
claim47	CLAIM47
common	TECH
complex_record01	C RECORD
complex_record02	C-RECORD

Problem Test Name	Problem Source File Name
complex_record03	C RECORD
complex_record04	C RECORD
complex_record05	C RECORD
complex_record06	C RECORD
complex_record07	C_RECORD
complex_record08	C_RECORD
complex_record09	C_RECORD
consistent1	CON
consistent2	CON
consistent3	CON
consistent4	CON
consistent5	CON
consistent6	CON
consistent7	CON
crc0	CRC
crc1	CRC
crc2	CRC
crc3	CRC
crc4	CRC
cse1	CSE
cse2	CSE
cse3	CSE
cse4	CSE
cse5	CSE
cse6	CSE
cse7	CSE
cse8	CSE
cse9	CSE
cse10	CSE
d library 1	D LIB
d library 2	D LIB
d library 3	D LIB
d_library_5	D_LIB

Problem Test Name	Problem Source File Name
d library 6	D LIB
d library 7	D LIB
d library 8	D LIB
dead	TECH
delay1	DELAYS
delay2	DELAYS
delay3	DELAYS
delay4	DELAYS
delay5	DELAYS
delay6	DELAYS
delay7	DELAYS
delay8	DELAYS
delay9	DELAYS
delay10	DELAYS
delay11	DELAYS
delay12	DELAYS
delay13	DELAYS
delay14	DELAYS
delay_abort1	D_ABORT
delay_abort2	D_ABORT
delay_zero0	DELAY0
delay_zero1	DELAY1.3
delay_zero2	DELAY1.3
delay_zero3	DELAY1.3
delay_zero4	DELAY4.6
delay_zero5	DELAY4.6
delay_zero6	DELAY4.6
delay_zero6x	DELAY6X
delay_zero7	DELAY7
delay_zero8	DELAY8
des1	DES1
des2	DES2
des3	DES3

Problem Test Name	Problem Source File Name
des4	DES4
des4a	DES4
des5	DES5
des5a	DES5
des6	DES6
des6a	DES6
des7	DES7
des7a	DES7
dhrys1	WITHDRAWN
dhrys1.mod	DHRYS1
dhrys2	WITHDRAWN
dhrys2.mod	DHRYS2
dhrys3	WITHDRAWN
dhrys3 mod	DHRYS3
elab1	ELAB1
elab10	ELAB2
elab2	ELAB1
elab3	ELAB1
elab4	ELAB1
elab5	ELAB1
elab6	ELAB2
elab7	ELAB2
elab8	ELAB2
elab9	ELAB2
enum.io1	ENUM_IO
enum.io2	ENUM_IO
enum.io3	ENUM_IO
enum io4	ENUM IO
enum io5	ENUM IO
enum io6	ENUM IO
enum io7	ENUM IO
enum io8	ENUM IO2
enum.io9	ENUM_IO3

Problem Test Name	Problem Source File Name
ew	EW
filter1	FILTER
filter1i	FILTER
filter2	FILTER
filter2i	FILTER
filter3	FILTER
filter4	FILTER
firth1	FIRTH
firth1x	FIRTH
firth2	FIRTH
firth2x	FIRTH
firth2y	FIRTH
firth3	FIRTH
firth3x	FIRTH
firth4	FIRTH
firth4x	FIRTH
firth5	FIRTH
firth5v	FIRTH
firth5w	FIRTH
firth5x	FIRTH
firth5y	FIRTH
firth5z	FIRTH
firth6	FIRTH
firth6x	FIRTH
firth7	FIRTH
firth7x	FIRTH7X
fold	WITHDRAWN
fold1	FOLD
fold2	FOLD
fold3	FOLD
fold4	FOLD
fold5	FOLD
fold6	FOLD

Problem Test Name	Problem Source File Name
fold7	FOLD
fold8	FOLD
fold mod	TECH
forward euler1	SA8TEST
forward_euler2	SA8TEST
funcexcp	FUNCEXCP
gamm	GAMM
gamm2	GAMM2
heapify	CFA
idioms	TECH
inst1	INST
inst2	INST
inst3	INST
inst4	INST
inst5	INST
int 0	INT 0
int 1	INT 1
int 2	INT 2
int_3	INT_3
int_4	INT_4
int_5	INT_5
int_6	INT_6
int_7	INT_7
int_8	INT_8
int_9	INT_9
invar	TECH
io0	IOTEST1
io1	IOTEST1
io2	IOTEST1
io3	IOTEST1
io4	IOTEST1
io5	IOTEST1
io6	IOTEST1

Problem Test Name	Problem Source File Name
io7	IOTEST1
io8	IOTEST1
io9	IOTEST1
io10	IOTEST1
io11	IOTEST2
io12	IOTEST2
io13	IOTEST2
io14	IOTEST2
io15	IOTEST2
io16	IOTEST2
io17	IOTEST3
io18	IOTEST3
io19	IOTEST3
io20	IOTEST3
io21	IOTEST3
io22	IOTEST3
io23	IOTEST3
io24	IOTEST4
io25	IOTEST4
io26	IOTEST4
io27	IOTEST4
io28	IOTEST4
io29	IOTEST4
io30	IOTEST4
io_80_20_1	IO_80A
io_80_20_2	IO_80A
io_80_20_3	IO_80A
io 80 20 4	IO 80A
io 80 20 5	IO 80B
io 80 20 6	IO 80B
io 80 20 7	IO 80B
io 80 20 8	IO 80B
io_80_20_9	IO_80B

Problem Test Name	Problem Source File Name
io 80 20 10	IO 80B
io copy1	IO COPY
io copy2	IO COPY
io copy3	IO COPY
io copy4	IO COPY
io inter1	IO INTER
io inter2	IO INTER
io inter3	IO INTER
io mem1	IO MEM
io mem2	IO MEM
io mem3	IO MEM
io pattern1	IO PAT
io pattern2	IO PAT
io pattern3	IO PAT
io pattern4	IO PAT
io pattern5	IO PAT
io pattern6	IO PAT
io pattern7	IO PAT
io pattern8	IO PAT
io recur1	IO RECUR
io recur2	IO RECUR
io recur3	IO RECUR
io scan1	IO SCAN
io scan2	IO SCAN
io scan2x	IO SCAN
io scan3	IO SCAN
io scan4	IO SCAN
io scan5	IO SCAN
io scan6	IO SCAN
io scan7	IO SCAN
io scan8	IO SCAN
io scan11	IO SCAN3
io scan12	IO SCAN3

Problem Test Name	Problem Source File Name
io_scan13	IO_SCAN4
io_scan14	IO_SCAN4
io_scan15	IO_SCAN4
io_scan16	IO_SCAN5
io_scan17	IO_SCAN5
io_scan18	IO_SCAN5
io_unif1	IO_UNIF1
io_unif2	IO_UNIF1
io_unif3	IO_UNIF1
io_unif4	IO_UNIF1
io_unif5	IO_UNIF2
io_unif6	IO_UNIF2
iqsort	SORT
kalman	KALMAN
kernel1	KERNEL1
kernel2	KERNEL2
kernel3	KERNEL3
kernel4	KERNEL4
kernel5	KERNEL5
kernel6	KERNEL6
kernel7	KERNEL7
kernel8	KERNEL8
kernel9	KERNEL9
kernel10	KERNEL10
kernel11	KERNEL11
kernel12	KERNEL12
kernel13	KERNEL13
kernel14	KERNEL14
kernel15	KERNEL15
kernel16	KERNEL16
kernel16 goto	KERNEL16
kernel17	KERNEL17
kernel18	KERNEL18

Problem Test Name	Problem Source File Name
kernel19	KERNEL19
kernel20	KERNEL20
kernel21	KERNEL21
kernel22	KERNEL22
kernel23	KERNEL23
kernel24	KERNEL24
label	LABEL
loop0	LOOP0
loop1	LOOP1
loop2	LOOP2
loop3	LOOP3
loop4a	LOOP4A
loop4b	LOOP4B
loop4c	LOOP4C
loop5	LOOP5
loop6	LOOP6
loop7	LOOP7
loop8	LOOP8
loop9	LOOP9
loop10	LOOP10
loop11	LOOP11
loop12	LOOP12
loop13	LOOP13
loop14	LOOP14
loop15	LOOP15
loop16	LOOP16
loop17	LOOP17
lu	CFA
merge1	SORT
merge2	SORT
neural	NEURAL
pure1	PURE
pure2	PURE

Problem Test Name	Problem Source File Name
pure3	PURE
pure4	PURE
pure5	PURE
pure6	PURE
pure7	PURE
pure8	PURE
puzzle	HANSON
qsort1	SORT
qsort2	SORT
queens	WITHDRAWN
queens.mod	QUEENS
reclaim_collection_constrained	RECLAIM
reclaim_collection_unconstrained	RECLAIM
reclaim_global_heap_constrained	RECLAIM
reclaim_global_heap_unconstrained	RECLAIM
reed_solomon_0	REED
reed_solomon_1	REED
reed_solomon_2	REED
reed_solomon_3	REED
reed_solomon_4	REED
runge	CFA
s_library_1	S_LIB
s_library_2	S_LIB
s_library_3	S_LIB
s_library_5	S_LIB
s_library_6	S_LIB
s_library_7	S_LIB
s_library_8	S_LIB
search	HANSON
shell1	SORT
shell2	SORT
sieve	HANSON
simulate_bmbat	SIMULATE

Problem Test Name	Problem Source File Name
simulate emrpm	SIMULATE
simulate hmproto	SIMULATE
simulate qmpitch	SIMULATE
simulate_rcwfrdet	SIMULATE
simulate_umnav	SIMULATE
simulate_kmdump	SIMULATE
simulate_rmkeying	SIMULATE
slice1	SLICE
slice2	SLICE
slice3	SLICE
slice4	SLICE
slice5	SLICE
slice6	SLICE
slice7	SLICE
slice8	SLICE
ss0	S0000T14
ss1	S0000T14
ss2	S0000T14
ss2_mod1	S0000T14
ss2_mod2	S0000T14
ss3	S0000T14
ss4	S0000T14
ss5	S0000T14
ss6	S0000T14
ss7	S0000T14
ss8	S0000T14
ss8_mod	S0000T14
ss9	S0000T14
ss10	S0000T14
ss11	S0000T14
ss12	S0000T14
ss13	S0000T14
ss14	S0000T14

Problem Test Name	Problem Source File Name
ss15	S0015T29
ss16	S0015T29
ss17	S0015T29
ss18	S0015T29
ss19	S0015T29
ss20	S0015T29
ss21	S0015T29
ss22	S0015T29
ss23	S0015T29
ss24	S0015T29
ss25	S0015T29
ss26	S0015T29
ss27	S0015T29
ss28	S0015T29
ss29	S0015T29
ss30	S0030T44
ss31	S0030T44
ss32	S0030T44
ss33	S0030T44
ss34	S0030T44
ss35	S0030T44
ss36	S0030T44
ss37	S0030T44
ss38	S0030T44
ss39	S0030T44
ss40	S0030T44
ss41	S0030T44
ss41 mod	S0030T44
ss42	S0030T44
ss42 mod	S0030T44
ss43	S0030T44
ss44	S0030T44
ss45	S0045T59

Problem Test Name	Problem Source File Name
ss46	S0045T59
ss47	S0045T59
ss48	S0045T59
ss49	S0045T59
ss50	S0045T59
ss51	S0045T59
ss52	S0045T59
ss53	S0045T59
ss54	S0045T59
ss55	S0045T59
ss56	S0045T59
ss57	S0045T59
ss58	S0045T59
ss59	S0045T59
ss60	S0060T74
ss61	S0060T74
ss62	S0060T74
ss63	S0060T74
ss64	S0060T74
ss65	S0060T74
ss66	S0060T74
ss67	S0060T74
ss68	S0060T74
ss69	S0060T74
ss70	S0060T74
ss71	S0060T74
ss72	S0060T74
ss73	S0060T74
ss74	S0060T74
ss75	S0075T89
ss76	S0075T89
ss77	S0075T89
ss78	S0075T89

Problem Test Name	Problem Source File Name
ss79	S0075T89
ss80	S0075T89
ss81	S0075T89
ss82	S0075T89
ss83	S0075T89
ss84	S0075T89
ss85	S0075T89
ss86	S0075T89
ss87	S0075T89
ss88	S0075T89
ss89	S0075T89
ss90	S0090T04
ss91	S0090T04
ss92	S0090T04
ss93	S0090T04
ss94	S0090T04
ss95	WITHDRAWN
ss95 mod	S0090T04
ss96	WITHDRAWN
ss96_mod	S0090T04
ss97	WITHDRAWN
ss97_mod	S0090T04
ss98	WITHDRAWN
ss98_mod	S0090T04
ss99	S0090T04
ss100	S0090T04
ss101	S0090T04
ss102	S0090T04
ss103	S0090T04
ss104	S0090T04
ss105	S0105T19
ss106	S0105T19
ss107	S0105T19

Problem Test Name	Problem Source File Name
ss108	S0105T19
ss109	S0105T19
ss110	S0105T19
ss111	S0105T19
ss112	S0105T19
ss113	S0105T19
ss114	S0105T19
ss115	S0105T19
ss116	S0105T19
ss117	S0105T19
ss118	S0105T19
ss119	S0105T19
ss120	S0120T34
ss121	S0120T34
ss122	S0120T34
ss123	S0120T34
ss124	S0120T34
ss125	S0120T34
ss126	S0120T34
ss127	S0120T34
ss128	S0120T34
ss129	S0120T34
ss130	S0120T34
ss131	S0120T34
ss132	S0120T34
ss133	S0120T34
ss134	S0120T34
ss135	S0135T48
ss136	S0135T48
ss137	S0135T48
ss138	S0135T48
ss139	S0135T48
ss140	S0135T48

Problem Test Name	Problem Source File Name
ssl141	S0135T48
ssl142	S0135T48
ssl143	S0135T48
ssl144	S0135T48
ssl145	S0135T48
ssl146	S0135T48
ssl147	S0135T48
ssl148	S0135T48
ssl149	S0149T61
ssl150	S0149T61
ssl151	S0149T61
ssl152	S0149T61
ssl153	S0149T61
ssl154	S0149T61
ssl155	S0149T61
ssl156	S0149T61
ssl157	S0149T61
ssl158	S0149T61
ssl159	S0149T61
ssl160	S0149T61
ssl161	S0149T61
ssl162	S0162T67
ssl163	S0162T67
ssl164	S0162T67
ssl165	S0162T67
ssl166	S0162T67
ssl167	S0162T67
ssl168	S0168T75
ssl169	S0168T75
ssl170	S0168T75
ssl171	S0168T75
ssl172	S0168T75
ssl173	S0168T75

Problem Test Name	Problem Source File Name
ss174	S0168T75
ss175	S0168T75
ss176	S0176T82
ss177	S0176T82
ss178	S0176T82
ss179	S0176T82
ss180	S0176T82
ss181	S0176T82
ss182	S0176T82
ss183	S0183T97
ss184	S0183T97
ss185	S0183T97
ss186	S0183T97
ss187	S0183T97
ss188	S0183T97
ss189	S0183T97
ss190	S0183T97
ss191	S0183T97
ss192	S0183T97
ss193	S0183T97
ss194	S0183T97
ss195	S0183T97
ss196	S0183T97
ss197	S0183T97
ss198	S0198T12
ss199	S0198T12
ss200	S0198T12
ss201	S0198T12
ss202	S0198T12
ss203	S0198T12
ss204	S0198T12
ss205	S0198T12
ss206	S0198T12

Problem Test Name	Problem Source File Name
ss207	S0198T12
ss208	S0198T12
ss209	S0198T12
ss210	S0198T12
ss211	S0198T12
ss212	S0198T12
ss213	S0213T27
ss214	S0213T27
ss215	S0213T27
ss216	S0213T27
ss216_mod	S0213T27
ss217	S0213T27
ss218	S0213T27
ss219	S0213T27
ss219 mod	S0213T27
ss220	S0213T27
ss221	S0213T27
ss222	S0213T27
ss223	S0213T27
ss224	S0213T27
ss225	S0213T27
ss226	S0213T27
ss227	S0213T27
ss228	S0228T41
ss229	S0228T41
ss230	S0228T41
ss231	S0228T41
ss232	S0228T41
ss233	S0228T41
ss234	S0228T41
ss235	S0228T41
ss236	S0228T41
ss237	S0228T41

Problem Test Name	Problem Source File Name
ss238	S0228T41
ss239	S0228T41
ss240	S0228T41
ss241	S0228T41
ss242	S0242T50
ss243	S0242T50
ss244	S0242T50
ss245	S0242T50
ss246	S0242T50
ss247	S0242T50
ss248	S0242T50
ss249	S0242T50
ss250	S0242T50
ss251	S0251T51
ss252	S0252T52
ss253	S0253T53
ss254	S0254T57
ss255	S0254T57
ss256	S0254T57
ss257	S0254T57
ss258	S0258T72
ss259	S0258T72
ss260	S0258T72
ss261	S0258T72
ss262	S0258T72
ss263	S0258T72
ss264	S0258T72
ss265	S0258T72
ss266	S0258T72
ss267	S0258T72
ss268	S0258T72
ss269	S0258T72
ss270	S0258T72

Problem Test Name	Problem Source File Name
ss271	S0258T72
ss272	S0258T72
ss273	S0273T85
ss274	S0273T85
ss275	S0273T85
ss276	S0273T85
ss277	S0273T85
ss278	S0273T85
ss279	S0273T85
ss280	S0273T85
ss281	S0273T85
ss282	S0273T85
ss283	S0273T85
ss284	S0273T85
ss285	S0273T85
ss286	S0286T00
ss287	S0286T00
ss288	S0286T00
ss289	S0286T00
ss290	S0286T00
ss291	S0286T00
ss292	S0286T00
ss293	S0286T00
ss294	S0286T00
ss295	S0286T00
ss296	S0286T00
ss297	S0286T00
ss298	S0286T00
ss299	S0286T00
ss300	S0286T00
ss301	S0301T15
ss302	S0301T15
ss303	S0301T15

Problem Test Name	Problem Source File Name
ss304	S0301T15
ss305	S0301T15
ss306	S0301T15
ss307	S0301T15
ss308	S0301T15
ss309	S0301T15
ss310	S0301T15
ss311	S0301T15
ss312	S0301T15
ss313	S0301T15
ss314	S0301T15
ss315	S0301T15
ss316	S0316T30
ss317	S0316T30
ss318	S0316T30
ss319	S0316T30
ss320	S0316T30
ss321	S0316T30
ss322	S0316T30
ss323	S0316T30
ss324	S0316T30
ss325	S0316T30
ss326	S0316T30
ss327	S0316T30
ss328	S0316T30
ss329	S0316T30
ss330	S0316T30
ss331	S0331T45
ss332	S0331T45
ss333	S0331T45
ss334	S0331T45
ss335	S0331T45
ss336	S0331T45

Problem Test Name	Problem Source File Name
ss337	S0331T45
ss338	S0331T45
ss339	S0331T45
ss340	S0331T45
ss341	S0331T45
ss342	S0331T45
ss343	S0331T45
ss344	S0331T45
ss345	S0331T45
ss346	S0346T53
ss347	S0346T53
ss348	S0346T53
ss349	S0346T53
ss350	S0346T53
ss351	S0346T53
ss352	S0346T53
ss353	S0346T53
ss354	S0354T68
ss355	S0354T68
ss356	S0354T68
ss357	S0354T68
ss358	S0354T68
ss359	S0354T68
ss360	S0354T68
ss361	S0354T68
ss362	S0354T68
ss363	S0354T68
ss364	S0354T68
ss365	S0354T68
ss366	S0354T68
ss367	S0354T68
ss368	S0354T68
ss369	S0369T78

Problem Test Name	Problem Source File Name
ss370	S0369T78
ss371	S0369T78
ss372	S0369T78
ss373	S0369T78
ss374	S0369T78
ss375	S0369T78
ss376	S0369T78
ss377	S0369T78
ss378	S0369T78
ss379	S0379T93
ss380	S0379T93
ss381	S0379T93
ss382	S0379T93
ss383	S0379T93
ss384	S0379T93
ss385	S0379T93
ss385x	S0379T93
ss386	S0379T93
ss387	S0379T93
ss388	S0379T93
ss389	S0379T93
ss390	S0379T93
ss391	S0379T93
ss392	S0379T93
ss393	S0379T93
ss394	S0394T08
ss395	S0394T08
ss396	S0394T08
ss397	S0394T08
ss398	S0394T08
ss399	S0394T08
ss400	S0394T08
ss401	S0394T08

Problem Test Name	Problem Source File Name
ss402	S0394T08
ss403	S0394T08
ss404	S0394T08
ss405	S0394T08
ss406	S0394T08
ss407	S0394T08
ss408	S0394T08
ss409	S0409T23
ss410	S0409T23
ss411	S0409T23
ss412	S0409T23
ss413	S0409T23
ss414	S0409T23
ss415	S0409T23
ss416	S0409T23
ss417	S0409T23
ss418	S0409T23
ss419	S0409T23
ss420	S0409T23
ss421	S0409T23
ss422	S0409T23
ss423	S0409T23
ss424	S0424T38
ss425	S0424T38
ss426	S0424T38
ss427	S0424T38
ss428	S0424T38
ss429	S0424T38
ss430	S0424T38
ss431	S0424T38
ss432	S0424T38
ss433	S0424T38
ss434	S0424T38

Problem Test Name	Problem Source File Name
ss435	S0424T38
ss436	S0424T38
ss437	S0424T38
ss438	S0424T38
ss439	S0439T43
ss440	S0439T43
ss441	S0439T43
ss442	S0439T43
ss443	S0439T43
ss444	S0444T47
ss445	S0444T47
ss446	S0444T47
ss447	S0444T47
ss448	S0448T49
ss449	S0448T49
ss450	S0450T51
ss451	S0450T51
ss452	S0452T66
ss453	S0452T66
ss454	S0452T66
ss455	S0452T66
ss456	S0452T66
ss457	S0452T66
ss458	S0452T66
ss459	S0452T66
ss460	S0452T66
ss461	S0452T66
ss462	S0452T66
ss463	S0452T66
ss464	S0452T66
ss465	S0452T66
ss466	S0452T66
ss467	S0467T78

Problem Test Name	Problem Source File Name
ss468	S0467T78
ss469	S0467T78
ss470	S0467T78
ss471	S0467T78
ss472	S0467T78
ss473	S0467T78
ss474	S0467T78
ss475	S0467T78
ss476	S0467T78
ss477	S0467T78
ss478	S0467T78
ss479	S0479T88
ss480	S0479T88
ss481	S0479T88
ss482	S0479T88
ss483	S0479T88
ss484	S0479T88
ss485	S0479T88
ss486	S0479T88
ss487	S0479T88
ss488	S0479T88
ss489	S0489T99
ss490	S0489T99
ss491	S0489T99
ss492	S0489T99
ss493	S0489T99
ss494	S0489T99
ss495	S0489T99
ss496	S0489T99
ss497	S0489T99
ss498	S0489T99
ss499	S0489T99
ss500	S0500T12

Problem Test Name	Problem Source File Name
ss501	S0500T12
ss502	S0500T12
ss503	S0500T12
ss504	S0500T12
ss505	S0500T12
ss506	S0500T12
ss507	S0500T12
ss508	S0500T12
ss509	S0500T12
ss510	S0500T12
ss511	S0500T12
ss512	S0500T12
ss513	S0513T28
ss514	S0513T28
ss515	S0513T28
ss516	S0513T28
ss517	S0513T28
ss518	S0513T28
ss519	S0513T28
ss520	S0513T28
ss521	S0513T28
ss522	S0513T28
ss523	S0513T28
ss524	S0513T28
ss525	S0513T28
ss526	S0513T28
ss527	S0513T28
ss528	S0513T28
ss529	S0529T42
ss530	S0529T42
ss531	S0529T42
ss532	S0529T42
ss533	S0529T42

Problem Test Name	Problem Source File Name
ss534	S0529T42
ss535	S0529T42
ss536	S0529T42
ss537	S0529T42
ss538	S0529T42
ss539	S0529T42
ss540	S0529T42
ss541	S0529T42
ss542	S0529T42
ss542x	S0529T42
ss543	S0543T57
ss544	S0543T57
ss545	S0543T57
ss546	S0543T57
ss547	S0543T57
ss548	S0543T57
ss549	S0543T57
ss550	S0543T57
ss551	S0543T57
ss552	S0543T57
ss553	S0543T57
ss554	S0543T57
ss555	S0543T57
ss556	S0543T57
ss557	S0543T57
ss558	S0558T74
ss559	S0558T74
ss560	S0558T74
ss561	S0558T74
ss561x	S0558T74
ss562	S0558T74
ss563	S0558T74
ss564	S0558T74

Problem Test Name	Problem Source File Name
ss565	S0558T74
ss566	S0558T74
ss567	S0558T74
ss568	S0558T74
ss569	S0558T74
ss570	S0558T74
ss571	S0558T74
ss572	S0558T74
ss573	S0558T74
ss574	S0558T74
ss575	S0575T89
ss576	S0575T89
ss577	S0575T89
ss578	S0575T89
ss579	S0575T89
ss580	S0575T89
ss581	S0575T89
ss582	S0575T89
ss583	S0575T89
ss584	S0575T89
ss585	S0575T89
ss586	S0575T89
ss587	S0575T89
ss588	S0575T89
ss589	S0575T89
ss590	S0590T97
ss591	S0590T97
ss592	S0590T97
ss593	S0590T97
ss594	S0590T97
ss595	S0590T97
ss596	S0590T97
ss597	S0590T97

Problem Test Name	Problem Source File Name
ss598	S0598T05
ss599	S0598T05
ss600	S0598T05
ss601	S0598T05
ss602	S0598T05
ss603	S0598T05
ss604	S0598T05
ss605	S0598T05
ss606	S0606T12
ss607	S0606T12
ss608	S0606T12
ss609	S0606T12
ss610	S0606T12
ss611	S0606T12
ss612	S0606T12
ss613	S0613T15
ss614	S0613T15
ss615	S0613T15
ss616	S0616T30
ss617	S0616T30
ss618	S0616T30
ss619	S0616T30
ss620	S0616T30
ss621	S0616T30
ss622	S0616T30
ss623	S0616T30
ss624	S0616T30
ss625	S0616T30
ss626	S0616T30
ss627	S0616T30
ss628	S0616T30
ss629	S0616T30
ss630	S0616T30

Problem Test Name	Problem Source File Name
ss631	S0631T44
ss632	S0631T44
ss633	S0631T44
ss634	S0631T44
ss635	S0631T44
ss636	S0631T44
ss637	S0631T44
ss638	S0631T44
ss639	S0631T44
ss640	S0631T44
ss641	S0631T44
ss642	S0631T44
ss643	S0631T44
ss643x	S0631T44
ss644	S0631T44
ss645	S0645T51
ss646	S0645T51
ss647	S0645T51
ss648	S0645T51
ss649	S0645T51
ss650	S0645T51
ss651	S0645T51
ss652	S0652T66
ss653	S0652T66
ss654	S0652T66
ss655	S0652T66
ss656	S0652T66
ss657	S0652T66
ss658	S0652T66
ss659	S0652T66
ss660	S0652T66
ss661	S0652T66
ss662	S0652T66

Problem Test Name	Problem Source File Name
ss663	S0652T66
ss664	S0652T66
ss665	S0652T66
ss666	S0652T66
ss667	S0667T81
ss668	S0667T81
ss669	S0667T81
ss670	S0667T81
ss671	S0667T81
ss672	S0667T81
ss673	S0667T81
ss674	S0667T81
ss675	S0667T81
ss676	S0667T81
ss677	S0667T81
ss678	S0667T81
ss679	S0667T81
ss680	S0667T81
ss681	S0667T81
ss682	S0682T86
ss683	S0682T86
ss684	S0682T86
ss685	S0682T86
ss686x	S0682T86
ss686y	S0682T86
ss687	S0687T01
ss688	S0687T01
ss689	S0687T01
ss690	S0687T01
ss691	S0687T01
ss692	S0687T01
ss693	S0687T01
ss694	S0687T01

Problem Test Name	Problem Source File Name
ss695	S0687T01
ss696	S0687T01
ss697	S0687T01
ss698	S0687T01
ss699	S0687T01
ss700	S0687T01
ss701	S0687T01
ss702	S0702T16
ss703	S0702T16
ss704	S0702T16
ss705	S0702T16
ss706	S0702T16
ss707	S0702T16
ss708	S0702T16
ss709	S0702T16
ss710	S0702T16
ss711	S0702T16
ss712	S0702T16
ss713	S0702T16
ss714	S0702T16
ss715	S0702T16
ss716	S0702T16
ss717	S0717T20
ss718	S0717T20
ss719	S0717T20
ss720	S0717T20
ss721	X0721T23
ss722	X0721T23
ss723	X0721T23
ss724	WITHDRAWN
ss724 mod	S0724T40
ss725	WITHDRAWN
ss725.mod	S0724T40

Problem Test Name	Problem Source File Name
ss726	WITHDRAWN
ss726 mod	S0724T40
ss727	WITHDRAWN
ss727 mod	S0724T40
ss728	WITHDRAWN
ss728_mod	S0724T40
ss729	WITHDRAWN
ss729_mod	S0724T40
ss730	WITHDRAWN
ss730_mod	S0724T40
ss731	WITHDRAWN
ss731_mod	S0724T40
ss732	WITHDRAWN
ss732_mod	S0724T40
ss734	WITHDRAWN
ss734 mod	S0724T40
ss735	WITHDRAWN
ss735 mod	S0724T40
ss736	WITHDRAWN
ss736_mod	S0724T40
ss737	WITHDRAWN
ss737_mod	S0724T40
ss738	WITHDRAWN
ss738_mod	S0724T40
ss739	WITHDRAWN
ss739_mod	S0724T40
ss740	WITHDRAWN
ss740 mod	S0724T40
ss741	S0741T41
ss744	S0744T46
ss745	S0744T46
ss746	S0744T46
ss747	X0747T47

Problem Test Name	Problem Source File Name
ss748	S0748T50
ss749	S0748T50
ss750	S0748T50
ss751	S0751T57
ss752	S0751T57
ss753	S0751T57
ss754	S0751T57
ss755	S0751T57
ss756	S0751T57
ss757	S0751T57
ss758	S0758T60
ss759	S0758T60
ss760	S0758T60
ss761	S0761T63
ss762	S0761T63
ss763	S0761T63
ss764	S0764T78
ss765	S0764T78
ss766	S0764T78
ss767	S0764T78
ss768	S0764T78
ss769	S0764T78
ss770	S0764T78
ss771	S0764T78
ss772	S0764T78
ss773	S0764T78
ss774	S0764T78
ss775	S0764T78
ss776	S0764T78
ss777	S0764T78
ss778	S0764T78
ss779	S0779T88
ss780	S0779T88

Problem Test Name	Problem Source File Name
ss781	S0779T88
ss782	S0779T88
ss783	S0779T88
ss784	S0779T88
ss785	S0779T88
ss786	S0779T88
ss787	S0779T88
ss788	S0779T88
ss789	S0789T98
ss790	S0789T98
ss791	S0789T98
ss792	S0789T98
ss793	S0789T98
ss794	S0789T98
ss795	S0789T98
ss796	S0789T98
ss797	S0789T98
ss798	S0789T98
ss799	S0799T05
ss800	S0799T05
ss801	S0799T05
ss802	S0799T05
ss803	S0799T05
ss804	S0799T05
ss805	S0799T05
ss806	S0806T11
ss807	S0806T11
ss808	S0806T11
ss809	S0806T11
ss810	S0806T11
ss811	S0806T11
ssearch	SSEARCH
ssearch2	SSEARCH2

Problem Test Name	Problem Source File Name
strength	TECH
tak	TAK
target	CFA
task1	TASK1
task2	TASK2
task3	TASK3
task4	TASK4
task5	TASK5
task6	TASK6
task7	TASK7
task8	TASK8
task9	TASK9
task10	TASK10
task11	TASK11
task12	TASK12
task13	TASK13
task14	TASK14
task15	TASK15
task16	TASK16
task17	TASK17
task18	TASK18
task19	TASK19
task20	TASK20
task21	TASK21
task22	TASK22
task23	TASK23
task24	TASK24
task25	TASK25
task26	TASK26
task27	TASK27
task28	TASK28
task29	TASK29
task30	TASK30

Problem Test Name	Problem Source File Name
task31	TASK31
task32	TASK32
task33	TASK33
task34	TASK34
task34_delta	TASK34
task35	TASK35
task35_delta	TASK35
task36	TASK36
task37a	TASK37
task37b	TASK37
task38	TASK38
task39	TASK39
task40	TASK40
task41	TASK41
task42	TASK42
task43	TASK43
task44a	TASK44
task44b	TASK44
task45a	TASK45
task45b	TASK45
task46	TASK46
task46x	TASK46
task47	TASK47
task48	TASK48
task49	TASK49
task50	TASK50
task51	TASK51
task52	TASK52
task53	TASK53
task54	WITHDRAWN
task54 mod	TASK54
task55	WITHDRAWN
task55_mod	TASK55

Problem Test Name	Problem Source File Name
task56	TASK56
task57	TASK57
task58	TASK58
task59	TASK59
task60	TASK60
task_num_1	TASKSYS
task_num_5	TASKSYS
task_num_10	TASKSYS
task_num_15	TASKSYS
task_num_20	TASKSYS
task_num_25	TASKSYS
task_num_30	TASKSYS
task2_num_1	TASKSYS2
task2 num 5	TASKSYS2
task2 num 10	TASKSYS2
task2 num 15	TASKSYS2
task2 num 20	TASKSYS2
task2 num 25	TASKSYS2
task2_num_30	TASKSYS2
trie1	TRIE
trie2	TRIE
unreach	TECH
whet1	WHET1
whet2	WHET2
whet3	WHET3
whet4	WHET4

5.3 Appendix III, TAPE DESCRIPTION

This appendix contains a complete list of the 697 files on the delivery tape. These files use approximately 8.2 megabytes of disk storage.

Tape Description

ACKER2.A	ACTIVE.A	AIFRAME.A
ALIAS.A	ART1.A	ASMNUL.MAR
ASYNC1.A	ASYNC2.A	ASYNC3.A
ASYNC4.A	ASYNC5.A	AVL.A
A STAR.A	CFA.A	CIO.A
CIQSORT.A	CLAIM01.A	CLAIM02.A
CLAIM03.A	CLAIM04.A	CLAIM05.A
CLAIM06.A	CLAIM07.A	CLAIM08.A
CLAIM09.A	CLAIM10.A	CLAIM11.A
CLAIM12.A	CLAIM13.A	CLAIM14.A
CLAIM15.A	CLAIM16.A	CLAIM17.A
CLAIM18.A	CLAIM19.A	CLAIM20.A
CLAIM21.A	CLAIM22.A	CLAIM23.A
CLAIM24.A	CLAIM25.A	CLAIM26.A
CLAIM27.A	CLAIM28.A	CLAIM29.A
CLAIM30.A	CLAIM31.A	CLAIM32.A
CLAIM33.A	CLAIM34.A	CLAIM35.A
CLAIM36.A	CLAIM37.A	CLAIM38.A
CLAIM39.A	CLAIM40.A	CLAIM41.A
CLAIM42.A	CLAIM43.A	CLAIM44.A
CLAIM45.A	CLAIM46.A	CLAIM47.A
CLEANUP_DBG_FILES.COM	CMP.COM	CMP_1_DBG.COM
CMP_ACEC.UNX	CMP_BASE.UNX	CMP_CK.COM
CMP_DIFF_NAMES.COM	CMP_DIFF_NAMES.UNX	CMP_SP.COM
CMP_T.UNX	CMP_TOOLS.UNX	CMP_TS.UNX
CMP_TST_PR.UNX	COMPILE_ACEC.COM	COMPILE_AND_RUN.COM
COMPILE_BASELINE.COM	COMPILE_FORMAT.COM	COMPILE_TEST_SUITE.COM
COMPILE_TOOLS.COM	COMP_TIME.ADA	COMP_TIME.DUMMY
COMP_TIME.VAX	CON.A	CRC.A

Tape Description

CSE.A	C RECORD.A	DBG01.ADA
DBG01.COM	DBG01.T.ADA	DBG01.T.COM
DBG02.ADA	DBG02.COM	DBG02.T.ADA
DBG02.T.COM	DBG03.ADA	DBG03.COM
DBG03.T.ADA	DBG03.T.COM	DBG04.ADA
DBG04.COM	DBG04.T.ADA	DBG04.T.COM
DBG05.ADA	DBG05.COM	DBG06.ADA
DBG06.COM	DBG07.ADA	DBG07.COM
DBG07.T.ADA	DBG07.T.COM	DBG08.ADA
DBG08.COM	DBG09.ADA	DBG09.COM
DBG10.ADA	DBG10.COM	DBG11.ADA
DBG11.COM	DBG12A.ADA	DBG12A.COM
DBG12B.ADA	DBG13.T.ADA	DBG13.T.COM
DBG14.T.ADA	DBG14.T.COM	DBG15.ADA
DBG15.COM	DBG16.ADA	DBG16.COM
DBG16.1.COM	DBG16.2.COM	DBG17.ADA
DBG17.COM	DBG18.ADA	DBG18.COM
DBG19.ADA	DBG19.COM	DBG20.ADA
DBG20.COM	DBG20.T.ADA	DBG20.T.COM
DBG21.T.ADA	DBG21.T.COM	DBG22.ADA
DBG22.COM	DBG23.ADA	DBG23.COM
DBG24.ADA	DBG24.COM	DBG25.ADA
DBG25.COM	DBG26.T.ADA	DBG26.T.COM
DBG27.ADA	DBG27.COM	DBG28.ADA
DBG28.COM	DBG29.ADA	DBG29.COM
DBG_TEMPLATE.TXT	DBL.MATH.ADA	DBL.MATH.DEC
DBL.MATH.PORT	DBL.MATHTEST.ADA	DELAY0.A
DELAY1.3.A	DELAY4.6.A	DELAY6X.A
DELAY7.A	DELAY8.A	DELAYS.A
DEPTTEST.ADA	DES1.A	DES2.A
DES3.A	DES4.A	DES5.A
DES6.A	DES7.A	DHRYS1.A
DHRYS2.A	DHRYS3.A	DIAGCOMP.COM
DIAGFILL.COM	DIAGLINK.COM	DIAGNOS.COM
DIAGREAD.ADA	DIA.E01A.ADA	DIA.E01B.ADA
DIA.E02A.ADA	DIA.E02B.ADA	DIA.E03A.ADA

Tape Description

DIA E03B.ADA
DIA_E03E.ADA
DIA_E03H.ADA
DIA_E04B.ADA
DIA E05B.ADA
DIA_E07A.ADA
DIA_E08B.ADA
DIA E11A.ADA
DIA_E12C.ADA
DIA_E13A.ADA
DIA_E16A.ADA
DIA E17B.ADA
DIA_L02B.ADA
DIA_L03A.ADA
DIA_L04A.ADA
DIA R01A.ADA
DIA_R02C.ADA
DIA_R04A.ADA
DIA_R05B.ADA
DIA W02A.ADA
DIA_W03A.ADA
DIA_W04C.ADA
DIA W05B.ADA
DIA W07A.ADA
DIA_W10A.ADA
DIA_W13A.ADA
DIA W14B.ADA
DIA_W15C.ADA
DIA_W18A.ADA
D.LIB.A
ENUM IO.A
EW.A
FIRTH7X.A
FORMAT.COM
GAMM.A
GETADR.MAR

DIA E03C.ADA
DIA_E03F.ADA
DIA_E03I.ADA
DIA_E04C.ADA
DIA E05C.ADA
DIA_E07B.ADA
DIA_E09A.ADA
DIA E12A.ADA
DIA_E12D.ADA
DIA_E14A.ADA
DIA_E16B.ADA
DIA L01A.ADA
DIA_L02C.ADA
DIA_L03B.ADA
DIA_L05A.ADA
DIA R02A.ADA
DIA_R02D.ADA
DIA_R04B.ADA
DIA_W01A.ADA
DIA W02B.ADA
DIA_W04A.ADA
DIA_W04D.ADA
DIA W05C.ADA
DIA W08A.ADA
DIA_W11A.ADA
DIA_W13B.ADA
DIA W15A.ADA
DIA_W16A.ADA
DIA_W18B.ADA
ELAB1.A
ENUM IO2.A
FILTER.A
FOLD.A
FORMAT.UNX
GAMM2.A
GLOBAL.CLOCK

DIA E03D.ADA
DIA_E03G.ADA
DIA_E04A.ADA
DIA_E05A.ADA
DIA E06A.ADA
DIA_E08A.ADA
DIA_E10A.ADA
DIA E12B.ADA
DIA_E12E.ADA
DIA_E15A.ADA
DIA_E17A.ADA
DIA L02A.ADA
DIA_L02D.ADA
DIA_L03C.ADA
DIA_L05B.ADA
DIA R02B.ADA
DIA_R03A.ADA
DIA_R05A.ADA
DIA_W01B.ADA
DIA W02C.ADA
DIA_W04B.ADA
DIA_W05A.ADA
DIA W06A.ADA
DIA W09A.ADA
DIA_W12A.ADA
DIA_W14A.ADA
DIA W15B.ADA
DIA_W17A.ADA
D_ABORT.A
ELAB2.A
ENUM IO3.A
FIRTH.A
FORMAT.ADA
FUNCEXCP.A
GEN MATH.ADA
GLOBAL.CPU

Tape Description

GLOBAL.SIZ	HANSON.A	INCLUDE.ADA
INCLUDE.COM	INITTIME.CLOCK	INITTIME.CPU
INITTIME.SIZ	INITTIME.TXT	INST.A
INT_0.A	INT_1.A	INT_2.A
INT_3.A	INT_4.A	INT_5.A
INT_6.A	INT_7.A	INT_8.A
INT_9.A	IOTEST1.A	IOTEST2.A
IOTEST3.A	IOTEST4.A	IO_80A.A
IO_80B.A	IO_COPY.A	IO_INTER.A
IO_MEM.A	IO_PAT.A	IO_RECUR.A
IO_SCAN.A	IO_SCAN3.A	IO_SCAN4.A
IO_SCAN5.A	IO_UNIF1.A	IO_UNIF2.A
KALMAN.A	KERNEL1.A	KERNEL10.A
KERNEL11.A	KERNEL12.A	KERNEL13.A
KERNEL14.A	KERNEL15.A	KERNEL16.A
KERNEL17.A	KERNEL18.A	KERNEL19.A
KERNEL2.A	KERNEL20.A	KERNEL21.A
KERNEL22.A	KERNEL23.A	KERNEL24.A
KERNEL3.A	KERNEL4.A	KERNEL5.A
KERNEL6.A	KERNEL7.A	KERNEL8.A
KERNEL9.A	LABEL.A	LF.SSA
LIB.COM	LIB01.ADA	LIB01.COM
LIB02.ADA	LIB02.COM	LIB03.COM
LIB03A.ADA	LIB03B2.ADA	LIB03_1.ADA
LIB03_2.ADA	LIB03_B1.ADA	LIB04.COM
LIB04_A1.ADA	LIB04_A2.ADA	LIB04_A3.ADA
LIB04_A4.ADA	LIB04_A5.ADA	LIB04_B1.ADA
LIB04_B2.ADA	LIB04_B3.ADA	LIB04_B4.ADA
LIB04_B5.ADA	LIB04_C.ADA	LIB05.ADA
LIB05.COM	LIB06.COM	LIB07.ADA
LIB07.COM	LIB08.COM	LIB08A.ADA
LIB08B.ADA	LIB08C.ADA	LIB08D.ADA
LIB08E.ADA	LIB08F.ADA	LIB08G.ADA
LIB08H.ADA	LIB08I.ADA	LIB08K0.ADA
LIB08K1.ADA	LIB08K2.ADA	LIB08K3.ADA
LIB08K4.ADA	LIB08L1.ADA	LIB08L2.ADA

Tape Description

LIB08L3.ADA	LIB08L4.ADA	LIB08M1.ADA
LIB08M2.ADA	LIB08M3.ADA	LIB08M4.ADA
LIB09.ADA	LIB09.COM	LIB10.COM
LIB10A.ADA	LIB10B.ADA	LIB11.ADA
LIB11.COM	LIB12.COM	LIB13.ADA
LIB13.COM	LIB14.COM	LIB14D.COM
LIB14P.COM	LIB14P1.ADA	LIB14P10.ADA
LIB14P2.ADA	LIB14P3.ADA	LIB14P4.ADA
LIB14P5.ADA	LIB14P6.ADA	LIB14P7.ADA
LIB14P8.ADA	LIB14P9.ADA	LIB14S1.COM
LIB14S1A.ADA	LIB14S1B.ADA	LIB14S1C.ADA
LIB14S2.COM	LIB14S2A.ADA	LIB14S2B.ADA
LIB14S2C.ADA	LIB15.ADA	LIB15.COM
LIB16.ADA	LIB16.COM	LIB17.ADA
LIB17.COM	LIB18.ADA	LIB18.COM
LIB19.ADA	LIB19.COM	LIB TEMPLATE.TXT
LOOP0.A	LOOP1.A	LOOP10.A
LOOP11.A	LOOP12.A	LOOP13.A
LOOP14.A	LOOP15.A	LOOP16.A
LOOP17.A	LOOP2.A	LOOP3.A
LOOP4A.A	LOOP4B.A	LOOP4C.A
LOOP5.A	LOOP6.A	LOOP7.A
LOOP8.A	LOOP9.A	MATH.ADA
MATH.DEC	MATH.PORT	MATHTEST.ADA
MATH_DEPENDENT.DEC	MATH_DEPENDENT.PORT	
MEDIAN.ADA	MEDIAN.COM	MEDIAN.UNX
MED DATA CONSTRUCTOR.ADA		MED DATA CONSTRUCTOR.COM
MED_DATA_CONSTRUCTOR.UNX		MSC.ADA
NEURAL.A	OPT.SSA	PREPARE_DBG_DIR.COM
PROPOSAL.STY	PURE.A	QUEENS.A
RAN16.ADA	RAN32.ADA	READ2.TEX
RECLAIM.A	REED.A	RTS.SSA
RUN_ACEC.COM	RUN_ACEC.UNX	RUN_TEST_PROGRAMS.COM
RUN_TST_PR.UNX	S0000T14.A	S0015T29.A
S0030T44.A	S0045T59.A	S0060T74.A
S0075T89.A	S0090T04.A	S0105T19.A

Tape Description

S0120T34.A	S0135T48.A	S0149T61.A
S0162T67.A	S0168T75.A	S0176T82.A
S0183T97.A	S0198T12.A	S0213T27.A
S0228T41.A	S0242T50.A	S0251T51.A
S0252T52.A	S0253T53.A	S0254T57.A
S0258T72.A	S0273T85.A	S0286T00.A
S0301T15.A	S0316T30.A	S0331T45.A
S0346T53.A	S0354T68.A	S0369T78.A
S0379T93.A	S0394T08.A	S0409T23.A
S0424T38.A	S0439T43.A	S0444T47.A
S0448T49.A	S0450T51.A	S0452T66.A
S0467T78.A	S0479T88.A	S0489T99.A
S0500T12.A	S0513T28.A	S0529T42.A
S0543T57.A	S0558T74.A	S0575T89.A
S0590T97.A	S0598T05.A	S0606T12.A
S0613T15.A	S0616T30.A	S0631T44.A
S0645T51.A	S0652T66.A	S0667T81.A
S0682T86.A	S0686T86.A	S0687T01.A
S0702T16.A	S0717T20.A	S0721T23.A
S0724T40.A	S0741T41.A	S0744T46.A
S0747T47.A	S0748T50.A	S0751T57.A
S0758T60.A	S0761T63.A	S0764T78.A
S0779T88.A	S0789T98.A	S0799T05.A
S0806T11.A	SA8TEST.A	SETUP DBG.COM
SETUP_TEST_PROGRAMS.COM		SIMULATE.A
SIZE.ADA	SIZE.DUMMY	SIZE.VAX
SLICE.A	SORT.A	SPACE0.ADA
SPACE1.ADA	SPACER.ADA	SPACES.ADA
SSA.ADA	SSA.TXT	SSEARCH.A
SSEARCH2.A	STARTIME.CLOCK	STARTIME.CPU
STARTIME.SIZ	STARTIME.TXT	STOPTIME0.CLOCK
STOPTIME0.CPU	STOPTIME0.SIZ	STOPTIME0.TXT
STOPTIME2.CLOCK	STOPTIME2.CPU	STOPTIME2.SIZ
STOPTIME2.TXT	STYLE.SSA	SYSNAMES.TXT
S LIB.A	TAK.A	TASK1.A
TASK10.A	TASK11.A	TASK12.A

Tape Description

TASK13.A	TASK14.A	TASK15.A
TASK16.A	TASK17.A	TASK18.A
TASK19.A	TASK2.A	TASK20.A
TASK21.A	TASK22.A	TASK23.A
TASK24.A	TASK25.A	TASK26.A
TASK27.A	TASK28.A	TASK29.A
TASK3.A	TASK30.A	TASK31.A
TASK32.A	TASK33.A	TASK34.A
TASK35.A	TASK36.A	TASK37.A
TASK38.A	TASK39.A	TASK4.A
TASK40.A	TASK41.A	TASK42.A
TASK43.A	TASK44.A	TASK45.A
TASK46.A	TASK47.A	TASK48.A
TASK49.A	TASK5.A	TASK50.A
TASK51.A	TASK52.A	TASK53.A
TASK54.A	TASK55.A	TASK56.A
TASK57.A	TASK58.A	TASK59.A
TASK6.A	TASK60.A	TASK7.A
TASK8.A	TASK9.A	TASKSYS.A
TASKSYS2.A	TECH.A	TEMPLATE.DIA
TESTCAL1.ADA	TESTCAL2.ADA	TIME.ADA
TIME.DUMMY	TIME.VAX	TRIE.A
USER2.TEX	VDD2.TEX	WHET1.A
WHET2.A	WHET3.A	WHET4.A
X0721T23.A	X0747T47.A	

5.4 Appendix IV, QUARANTINED TEST PROBLEMS

This appendix contains a list of test problems which fail on some systems for various reasons, and a list of test problems which fail for system dependent reasons.

Problem Name	Number of Systems Failing
A_STAR	1
ACKER1	1
ACKER2	1
ACTIVATION1	2
ACTIVATION2	2
ALIAS1	1
ALIAS2	1
ALIAS3	2
ALIAS4	1
ALIAS5	1
ALIAS5X	1
ALIAS6	1
ALIAS6X	1
ALIAS7	1
ALIAS7X	1
ALIAS8	1
ALIAS8X	1
ALIAS9	1
ALIAS10	1
ALIAS11	1
ALIAS12	1
ALIAS13	3
ALIAS14	3
ALIAS15	3
ALIAS16	3
ASYNC1	1
ASYNC3	1
ASYNC5	2
AVL_0	2
AVL_1	2
AVL_2	2
AVL_3	2
AVL_4	2
AVL_5	2
AVL_6	2
AVL_7	2

Problem Name	Number of Systems Failing
AVL_8	2
AVL_9	2
AVL_10	2
AVL_11	2
BSORT1	1
BSORT2	1
CAT1	1
CAT2	1
CAT3	1
CIO1	1
CIO2	1
CIO3	1
CIO4	1
CIO5	1
CIO6	1
CIO7	1
CIO8	1
CIO9	1
CIO10	1
CIO11	1
CIO12	1
CIO13	1
CIO14	1
CLAIM01	3
CLAIM02	3
CLAIM03	3
CLAIM04	3
CLAIM05	3
CLAIM06	3
CLAIM07	3
CLAIM08	3
CLAIM09	2
CLAIM10	4

Problem Name	Number of Systems Failing
CLAIM11	3
CLAIM12	4
CLAIM13	4
CLAIM14	3
CLAIM15	3
CLAIM16	3
CLAIM17	2
CLAIM18	3
CLAIM19	3
CLAIM20	2
CLAIM21	2
CLAIM22	2
CLAIM23	2
CLAIM24	3
CLAIM25	2
CLAIM26	1
CLAIM27	3
CLAIM28	2
CLAIM29	2
CLAIM30	2
CLAIM31	1
CLAIM32	1
CLAIM33	1
CLAIM34	2
CLAIM35	2
CLAIM36	4
CLAIM37	4
CLAIM38	3
CLAIM39	1
CLAIM40	1
CLAIM41	2
CLAIM42	1
CLAIM43	1
CLAIM44	1
CLAIM45	2
CLAIM46	2
CLAIM47	2

Problem Name	Number of Systems Failing
COMMON	2
COMPLEX_RECORD01	2
COMPLEX_RECORD02	2
COMPLEX_RECORD03	2
COMPLEX_RECORD04	3
COMPLEX_RECORD05	2
COMPLEX_RECORD06	2
COMPLEX_RECORD07	2
COMPLEX_RECORD08	3
COMPLEX_RECORD09	3
CONSISTENT1	1
CONSISTENT2	1
CONSISTENT3	1
CONSISTENT4	1
CONSISTENT5	1
CONSISTENT6	1
CONSISTENT7	1
CRC0	3
CRC1	2
CRC2	3
CRC3	3
CRC4	3
D_LIBRARY_1	2
D_LIBRARY_2	2
D_LIBRARY_3	2
D_LIBRARY_5	2
D_LIBRARY_6	2
D_LIBRARY_7	2
D_LIBRARY_8	2
DEAD	2
DELAY1	1
DELAY2	1
DELAY3	1
DELAY_ABORT1	2
DELAY_ZERO0	1
DELAY_ZERO6X	1

Problem Name	Number of Systems Failing
DES1	3
DES2	1
DES3	1
DES4	1
DES4A	1
DES5	3
DES5A	2
DES6	3
DES6A	2
DES7	3
DES7A	3
ELAB1	1
ELAB2	1
ELAB3	1
ELAB4	1
ELAB5	1
ELAB6	1
ELAB7	1
ELAB8	1
ELAB9	1
ELAB10	1
ENUM.IO8	1
ENUM.IO9	2
EW	2
FOLD.MOD	2
FUNCEXCP	1
IDIOMS	2
INST1	1
INST2	1
INST3	1
INST4	1
INST5	1
INVAR	2

Problem Name	Number of Systems Failing
IO0	1
IO1	1
IO2	1
IO3	1
IO4	1
IO5	1
IO6	1
IO7	2
IO8	2
IO9	2
IO10	2
IO11	1
IO12	1
IO13	1
IO14	1
IO15	1
IO16	1
IO17	1
IO18	1
IO19	1
IO20	1
IO21	1
IO22	1
IO23	1
IO_80_20_5	1
IO 80 20 6	1
IO_80_20_7	1
IO_80_20_8	1
IO_80_20_9	1
IO 80 20 10	1
IO_COPY1	1
IO_COPY2	1
IO_COPY3	1
IO_COPY4	1
IO_INTER1	1
IO INTER2	1
IO_INTER3	1

Problem Name	Number of Systems Failing
IO_MEM1	1
IO_MEM2	1
IO MEM3	1
IO PATTERN1	1
IO_PATTERN2	1
IO_PATTERN3	1
IO PATTERN4	1
IO_PATTERN5	1
IO_PATTERN6	1
IO_PATTERN7	1
IO PATTERN8	1
IO RECUR1	1
IO RECUR2	1
IO_RECUR3	1
IO_SCAN1	1
IO_SCAN2	1
IO_SCAN2X	1
IO_SCAN3	1
IO_SCAN4	1
IO_SCAN5	1
IO_SCAN6	1
IO_SCAN7	1
IO_SCAN8	1
IO_UNIF1	2
IO_UNIF2	2
IO UNIF3	2
IO_UNIF4	2
IO_UNIF5	2
IO_UNIF6	2
IQSORT	1
LOOP0	1
LOOP1	1
LOOP4A	1
LOOP4B	1
LOOP4C	1
LOOP5	1
LOOP10	1

Problem Name	Number of Systems Failing
LOOP11	1
LOOP17	1
MERGE1	1
MERGE2	1
PUZZLE	1
QSORT1	1
QSORT2	1
RECLAIM_COLLECTION_CONSTRAINED	2
RECLAIM_COLLECTION_UNCONSTRAINED	2
RECLAIM_GLOBAL_HEAP_CONSTRAINED	1
RECLAIM GLOBAL HEAP UNCONSTRAINED	1
REED_SOLOMON_0	3
REED_SOLOMON_1	2
REED_SOLOMON_2	2
REED_SOLOMON_3	2
REED_SOLOMON_4	2
S_LIBRARY_1	1
S_LIBRARY_2	1
S_LIBRARY_3	1
S_LIBRARY_5	1
S_LIBRARY_6	2
S_LIBRARY_7	1
S_LIBRARY_8	1
SEARCH	1
SHELL1	1
SHELL2	1
SIEVE	1
SLICE1	1
SLICE2	1
SLICE3	1
SLICE4	1
SLICE5	1
SLICE6	1
SLICE7	1
SLICE8	1

Problem Name	Number of Systems Failing
SS60 .. ss74	1
SS162 .. ss167	1
SS213 .. ss227	1
SS228 .. ss241	1
SS242 .. ss250	1
SS250	2
SS253	1
SS254 .. ss257	1
SS316 .. ss330	1
SS331 .. ss345	1
SS346 .. ss353	1
SS369 .. ss378	1
SS458 .. ss466	1
SS500 .. ss512	1
SS513 .. ss528	1
SS537	1
SS539	2
SS652 .. ss666	1
SS682 .. ss686	1
SS687 .. ss701	3
SS702 .. ss716	2
SS717 .. ss720	1
SS721	2
SS722	1
SS741	3
SS764 .. ss780	1
STRENGTH	2

Problem Name	Number of Systems Failing
TASK1	1
TASK2	1
TASK4	1
TASK5	1
TASK6	2
TASK7	2
TASK8	1
TASK9	1
TASK25	1
TASK25	1
TASK30	1
TASK31	1
TASK32	1
TASK35	2
TASK35.DELTA	1
TASK37A	2
TASK37B	1
TASK40	1
TASK44A	2
TASK44B	2
TASK45A	2
TASK45B	3
TASK46	2
TASK46X	2
TASK47	1
TASK49	2
TASK50	2
TASK51	2
TASK52	2
TASK53	2
TASK54.MOD	3
TASK55 MOD	3
TASK56	2
TASK57	2
TASK58	1
TASK59	2
TASK60	1

Problem Name	Number of Systems Failing
TASK_NUM.1	2
TASK NUM 5	2
TASK NUM 10	3
TASK_NUM.15	2
TASK_NUM.20	3
TASK_NUM.25	2
TASK_NUM.30	2
TASK2_NUM.1	3
TASK2_NUM.5	2
TASK2 NUM 10	2
TASK2_NUM.15	2
TASK2_NUM.20	2
TASK2_NUM.25	2
TASK2 NUM 30	2
TRIE1	2
TRIE2	2
UNREACH	2

This part of Appendix IV contains a list of test problems which have been observed to fail on some systems for system dependent reasons.

Problem Name	Number of Systems Failing
ASYNC2	4
ASYNC4	3
AUTO	1
BMT	1
DELAY4	1
DELAY5	1
DELAY6	1
DELAY7	1
DELAY8	1
DELAY9	1
DELAY10	1
DELAY11	1
DELAY12	1
DELAY13	1
DELAY14	1
DELAY_ABORT2	1
DELAY_ZERO1	1
DELAY_ZERO2	1
DELAY_ZERO3	1
DELAY_ZERO4	1
DELAY_ZERO5	1
DELAY_ZERO6	1
ENUM_IO1	1
ENUM_IO2	1
ENUM_IO3	1
ENUM_IO4	1
ENUM_IO5	1
ENUM_IO6	1
ENUM_IO7	1
FORWARD_EULER1	1
FORWARD_EULER2	1
HEAPIFY	1

Problem Name	Number of Systems Failing
INT_0	1
INT_1	1
INT_2	1
INT_3	1
INT_4	1
INT_5	1
INT_6	1
INT_7	1
INT_8	1
INT_9	1
IO24	1
IO25	1
IO26	1
IO27	1
IO28	1
IO29	1
IO30	1
IO_80_20_1	1
IO_80_20_2	1
IO_80_20_3	1
IO_80_20_4	1
IO_80_20_5	1
IO_80_20_6	1
IO_80_20_7	1
IO_80_20_8	1
IO_80_20_9	1
IO_80_20_10	1
IO_SCAN11	1
IO_SCAN12	1
IO_SCAN13	1
IO_SCAN14	1
IO_SCAN15	1
IO_SCAN16	1
IO_SCAN17	1
IO_SCAN18	1
KALMAN	3
KERNEL1 .. kernel24	1

Problem Name	Number of Systems Failing
LOOP7	1
LOOP8	1
LU	1
NEURAL	1
RUNGE	1
SIMULATE_BMBAT	2
SIMULATE EMRPM	2
SIMULATE HMPROTO	2
SIMULATE_QMPITCH	2
SIMULATE_RCWFRDET	2
SIMULATE_UMNAV	2
SIMULATE_KMDUMP	2
SIMULATE_RMKEYING	2
SS0 .. ss14	1
SS15 .. ss29	1
SS30 .. ss44	1
SS45 .. ss59	1
SS258 .. ss272	1
SS273 .. ss285	1
SS286 .. ss300	1
SS301 .. ss315	1
SS394 .. ss408	1
SS409 .. ss423	1
SS424 .. ss438	1
SS439 .. ss443	1
SS558 .. ss574	1

Problem Name	Number of Systems Failing
SS575 .. ss589	1
SS590 .. ss597	1
SS616 .. ss630	1
SS631 .. ss644	1
SS645 .. ss651	1
SS723	4
SS724.MOD .. SS740.MOD	1
SS747	3
SS806 .. ss811	1
TARGET	1
TASK48	1
WHET1	2
WHET2	2
WHET3	2
WHET4	2

5.5 Appendix V, ACEC KEYWORD INDEX - 1

This appendix contains a list of primary purposes (with LRM references) and their associated test problems, as well as secondary, and incidental purposes, and comparison tests.

access.operations 3.8.2
Primary : ss154, ss155, ss256, ss257, ss648, ss746 (ss744..ss745),
ss748, ss805
Secondary : reclaim_collection_constrained,
reclaim_collection_unconstrained,
reclaim_global_heap_constrained,
reclaim_global_heap_unconstrained, ss161, ss162, ss163,
ss164, ss165, ss166, ss167, ss739_mod, trie1, trie2

application.ai 1.1.2
Primary : a_star, neural

application.avionics 1.1.2
Primary : arti_asum, arti_atan2, arti_cos, arti_fmod,
arti_ifpm_control, arti_ifpm_init, arti_ifpm_io,
arti_ifpm_rotors, arti_nairini, arti_nscni, arti_nutmini,
arti_sin, ew, forward_euler1, forward_euler2

application.avl_tree 1.1.2
Primary : avl_0, avl_1, avl_2, avl_3, avl_4, avl_5, avl_6, avl_7,
avl_8, avl_9, avl_10, avl_11

application.cyclic_redundancy_check 1.1.2
Primary : crc0, crc1, crc2, crc3, crc4

application.data_encryption_standard 1.1.2
Primary : des1, des2, des3, des4, des4a, des5, des5a, des6, des6a,
des7, des7a

application.error_correcting_code 1.1.2
Primary : reed_solomon_0, reed_solomon_1, reed_solomon_2,
reed_solomon_3, reed_solomon_4

application.filter 1.1.2
Primary : filter1, filter1i, filter2, filter2i, filter3, filter4

application.integration 1.1.2
Primary : ss398, ss402

application.kalman_filter 1.1.2
Primary : kalman

application.lag_filter 1.1.2
Primary : ss397, ss401

application.polynomial_coding_style 1.1.2
Primary : ss120, ss121, ss122, ss123

application.simulation 1.1.2
 Primary : simulate_bmbat, simulate_emrpm, simulate_hmproto,
 simulate_qmpitch, simulate_rcwfrdet, simulate_umnav,
 simulate_kmdump, simulate_rmkeying
 application.symmetric_deadzone 1.1.2
 Primary : ss399, ss403
 application.symmetric_limiter 1.1.2
 Primary : ss400, ss404
 application.trie 1.1.2
 Primary : trie1, trie2
 array.aggregates 4.3.1
 Primary : ss775, ss778
 Secondary : ss764, ss765, ss766, ss767, ss768
 array.constraints 3.6.1
 Primary : ss596 (ss597)
 array.dynamic 3.6
 Primary : ss419 (ss420)
 array.operations 3.6.2
 Primary : cat1, cat2, cat3, ss17, ss18, ss19, ss57, ss77, ss78, ss79,
 ss80, ss81, ss301, ss645, ss646, ss647, ss758, ss759, ss760,
 ss761, ss762, ss763, ss774, ss776, ss777
 Secondary : claim18, claim19, claim20, claim21, claim22, ss53, ss54,
 ss55, ss58, ss75, ss76, ss120, ss168, ss169, ss170, ss172,
 ss173, ss174, ss175, ss192, ss193, ss194, ss235, ss243,
 ss246, ss258, ss259, ss284, ss285, ss309, ss388, ss429,
 ss430, ss511, ss512, ss518, ss519, ss520, ss553, ss554
 Incidental : ss405, ss406, ss409, ss410, ss411, ss419, ss420, ss428,
 ss432, ss433, ss434, ss435, ss436, ss437, ss438, ss439,
 ss442, ss443, ss477, ss508, ss509, ss516, ss517, ss535,
 ss536, ss541, ss542, ss542x, ss545, ss557, ss562, ss596,
 ss597, ss648, ss652, ss653, ss654, ss655, ss656, ss657,
 ss658, ss659, ss660, ss661, ss662, ss663, ss664, ss665,
 ss666, ss667, ss668, ss669, ss670, ss671, ss672, ss673,
 ss674, ss675, ss676, ss677, ss678, ss679, ss680, ss681,
 ss687, ss688, ss689, ss690, ss691, ss692, ss693, ss694,
 ss695, ss696, ss697, ss698, ss699, ss700, ss701, ss702,
 ss703, ss704, ss705, ss706, ss707, ss708, ss709, ss710,
 ss711, ss712, ss713, ss714, ss715, ss716, ss731_mod,
 ss732_mod, ss734_mod, ss735_mod, ss749, ss750

boolean.arrays.packed 4.5
 Primary : ss337, ss338, ss339, ss340, ss341, ss342, ss343, ss344,
 ss345, ss347, ss348, ss349, ss524, ss525, ss526, ss764,
 ss765, ss766, ss767, ss768 (ss769..ss773)
 Incidental : ss346, ss353, ss500, ss501, ss502, ss506
 boolean.arrays.unpacked 4.5
 Primary : ss326, ss327, ss328, ss329, ss330, ss331, ss332, ss333,
 ss334, ss336, ss351, ss352
 Incidental : ss346, ss353, ss486
 boolean.expressions 4.5
 Primary : ss72, ss101, ss177, ss228, ss229, ss486, ss487, ss488, ss489,
 ss492, ss499, ss686x, ss686y
 Secondary : ss73, ss74, ss176, ss227, ss230, ss231, ss232, ss280, ss326,
 ss327, ss329, ss330, ss331, ss332, ss333, ss334, ss335,
 ss336, ss337, ss338, ss339, ss340, ss341, ss342, ss343,
 ss344, ss345, ss346, ss347, ss348, ss349, ss350, ss351,
 ss352, ss353, ss500, ss501, ss502
 Incidental : ss145, ss146, ss147, ss314, ss315, ss316, ss317, ss318,
 ss323, ss464, ss598, ss599, ss602, ss604, ss805
 boolean.record 3.5.3
 Primary : ss682, ss683, ss684, ss685, ss717, ss718, ss719, ss720
 classical.ackermann's 1.1.2
 Primary : acker1, acker2
 classical.cube_placing 1.1.2
 Primary : puzzle
 classical.dining_philosophers 1.1.2
 Primary : task7, task8, task9, task10, task25
 classical.dhrystone 1.1.2
 Primary : dhrys1_mod, dhrys2_mod, dhrys3_mod
 classical.eight_queens 1.1.2
 Primary : queens_mod
 classical.GAMM_measure 1.1.2
 Primary : gamm, gamm2
 classical.numerical.comp_fam_arch(CFA) 1.1.2
 Primary : auto, bmt, heapify, lu, runge, target
 classical.numerical.knuth_loops 1.1.2
 Primary : loop0, loop1, loop2, loop3, loop4a, loop4b, loop4c, loop5,
 loop6, loop7, loop8, loop9, loop10, loop11, loop12, loop13,
 loop14, loop15, loop16, loop17

classical.numerical.livermore_loops 1.1.2
 Primary : kernel1, kernel2, kernel3, kernel4, kernel5, kernel6,
 kernel7, kernel8, kernel9, kernel10, kernel11, kernel12,
 kernel13, kernel14, kernel15, kernel16 (kernel16_goto),
 kernel17, kernel18, kernel19, kernel20, kernel21, kernel22,
 kernel23, kernel24
 classical.prime_number 1.1.2
 Primary : sieve
 classical.search 1.1.2
 Primary : search, ssearch, ssearch2
 classical.sort 1.1.2
 Primary : bsort1, bsort2, ciqsort, iqsort, merge1, merge2, qsort1,
 qsort2, shell1, shell2
 classical.whetstone 1.1.2
 Primary : whet1, whet2, whet3, whet4
 consistency_check.timing_loop
 Primary : consistent1, consistent2, consistent3, consistent4,
 consistent5, consistent6, consistent7, ss769, ss770, ss771,
 ss772, ss773
 conversion.fixed 3.5.10
 Primary : ss107, ss108, ss466, ss467, ss721, ss722, ss723
 conversion.float 4.6
 Primary : ss2, ss2_mod1 (ss2_mod2), ss13, ss289, ss290
 Secondary : ss283
 conversion.integer 4.6
 Primary : ss8, ss8_mod, ss12, ss233, ss234, ss300, ss468
 Secondary : ss277, ss282, ss303
 conversion.null 4.6
 Primary : ss241
 conversion.packed_to_unpacked 4.6
 Primary : ss335, ss346, ss353
 conversion.unchecked_conversion 13.10.2
 Primary : ss259 (ss258), ss500, ss501, ss502, ss506
 conversion.unpacked_to_packed 4.6
 Primary : ss350

delay.problems	9.6
Primary	: delay1, delay2, delay3, delay4, delay5, delay6, delay7, delay8, delay9, delay10, delay11, delay12, delay13, delay14, delay_zero0, delay_zero1, delay_zero2, delay_zero3, delay_zero4, delay_zero5, delay_zero6 (delay_zero6x), delay_zero7, delay_zero8, ss455, ss458, ss459
Secondary	: async3, delay_abort1, delay_abort2
exception.handling	11.4
Primary	: funcexcp, ss379, ss380, ss381, ss382, ss383, ss384, ss527, ss528
Secondary	: ss543
Incidental	: cat3, claim12, claim13, claim14, claim15, claim19, claim21, claim22, claim38, claim46, claim47, ss598, ss599, ss602, ss604, ss638, ss741
exception.numeric_error	11.1
Primary	: ss313, ss369
exception.raise	11.3
Primary	: ss117, ss311, ss312
Secondary	: cat3, claim13, ss755, ss757
expression.abs	4.5.6
Primary	: ss29, ss30, ss266, ss293
Secondary	: ss368
Incidental	: ss431
expression.attributes	4.1
Primary	: ss246
expression.catenation	4.5.3
Primary	: ss113
expression.exponentiating	4.5
Primary	: ss191
Secondary	: ss15, ss16, ss21, ss50, ss51, ss65, ss66, ss188, ss213, ss216, ss216_mod, ss217, ss219, ss219_mod, ss279, ss291, ss304, ss305, ss306, ss307, ss643x
expression.parenthesis	4.5
Primary	: ss389, ss390, ss391, ss392, ss393, ss394, ss395, ss396
fixed.operations	3.5.10
Primary	: ss109, ss110, ss460, ss461, ss462, ss463, ss464, ss465

float.operations

3.5.8

Primary : ss1, ss3, ss4, ss5, ss6, ss211, ss286, ss287, ss288, ss302,
ss308, ss315, ss324, ss591 (ss592..ss594), ss643x

Secondary : ss20, ss21, ss22, ss23, ss24, ss25, ss59, ss60, ss61, ss62,
ss63, ss64, ss65, ss66, ss71, ss134, ss135, ss136, ss150,
ss216, ss216_mod, ss219, ss219_mod, ss220, ss256, ss257,
ss293, ss294, ss295, ss296, ss297, ss298, ss299, ss301,
ss314, ss316, ss317, ss318, ss323, ss389, ss390, ss391,
ss392, ss552, ss575, ss576, ss577, ss578, ss579, ss580,
ss581, ss582, ss583, ss585, ss588, ss589, ss590, ss595,
ss606, ss607, ss609, ss643, ss779, ss780, ss781, ss782,
ss783, ss784, ss785, ss786, ss787, ss788, ss789, ss790,
ss791, ss792, ss793, ss794, ss795, ss796, ss797, ss798

Incidental : ss67, ss68, ss69, ss70, ss120, ss121, ss122, ss123, ss141,
ss142, ss143, ss154, ss155, ss210, ss218, ss226, ss233,
ss234, ss262, ss263, ss291, ss292, ss304, ss305, ss306,
ss307, ss397, ss398, ss399, ss400, ss401, ss402, ss403,
ss404, ss406, ss407, ss413, ss414, ss415, ss416, ss417,
ss418, ss431, ss432, ss433, ss434, ss435, ss436, ss437,
ss442, ss443, ss444, ss448, ss450, ss454, ss467, ss485,
ss511, ss512, ss513, ss514, ss515, ss529, ss530, ss531,
ss532, ss533, ss534, ss535, ss536, ss547, ss548, ss549,
ss586, ss621, ss622, ss623, ss624, ss625, ss626, ss627,
ss628, ss629, ss630, ss631, ss632, ss633, ss645, ss646,
ss647, ss649, ss650, ss753, ss754, ss758, ss759, ss760,
ss761, ss762, ss763

generic.instanstiation

12.3

Primary : enum_io1, enum_io2, enum_io3, enum_io4, enum_io5, enum_io6,
enum_io7, enum_io8, enum_io9

generic.package

12.

Secondary : filter2, filter2i, ss806, ss807, ss808, ss809, ss810, ss811

generic.subprogram

12.

Primary : ss148, ss149 (ss151), ss150, ss478, ss621, ss622, ss623,
ss624, ss625, ss626, ss627, ss628, ss629, ss630, ss631

Secondary : filter1, filter1i

image

3.5.5

Primary : ss131

Secondary : claim17

Incidental : inst4, ss370

```

integer.bigint.operations          3.5.5
    Primary      : ss270, ss271, ss272, ss273, ss274, ss275, ss276, ss277,
                  ss278, ss280, ss282, ss283, ss284
integer.MOD                      3.5.5
    Primary      : ss102
    Secondary    : ss199
    Incidental   : ss446
integer.operations                3.5.5
    Primary      : ss7, ss9, ss10, ss11, ss46, ss201, ss202, ss203, ss268,
                  ss269, ss281, ss561, ss729_mod, ss744, ss745
    Secondary    : ss40, ss41, ss41_mod, ss42, ss42_mod, ss43, ss44, ss45, ss47,
                  ss48, ss49, ss50, ss51, ss52, ss56, ss137, ss189, ss195,
                  ss196, ss197, ss198, ss217, ss221, ss393, ss394, ss395,
                  ss396, ss503, ss550, ss551, ss556, ss560, ss566, ss567,
                  ss568, ss569, ss570, ss571, ss572, ss573, ss574, ss584,
                  ss608, ss610, ss611, ss753, ss754
    Incidental   : ss95_mod, ss96_mod, ss97_mod, ss98_mod, ss102, ss103, ss117,
                  ss129, ss130, ss131, ss138, ss139, ss140, ss190, ss191,
                  ss200, ss209, ss213, ss214, ss241, ss264, ss265, ss266,
                  ss267, ss364, ss366, ss367, ss369, ss372, ss373, ss374,
                  ss375, ss384, ss385x, ss386, ss423, ss424, ss425, ss426,
                  ss427, ss428, ss429, ss430, ss431, ss440, ss441, ss445,
                  ss446, ss447, ss449, ss451, ss466, ss468, ss469, ss470,
                  ss471, ss472, ss473, ss474, ss475, ss476, ss490, ss491,
                  ss500, ss501, ss502, ss506, ss507, ss511, ss512, ss558,
                  ss559, ss563, ss564, ss565, ss612, ss634, ss635, ss636,
                  ss637, ss638, ss639, ss640, ss651, ss652, ss752, ss755,
                  ss756, ss757, ss774, ss775, ss776, ss777, ss778
integer.REM                      4.5.5
    Primary      : ss103
    Secondary    : ss204
    Incidental   : ss276, ss362, ss363, ss447
interface.language.assembly       13.9
    Primary      : ss747

```

IO.direct 14.2
Primary : io11, io12, io13, io14, io15, io16, io_80_20_1, io_80_20_2,
io_80_20_3, io_80_20_4, io_80_20_5, io_80_20_6, io_80_20_7,
io_80_20_8, io_80_20_9, io_80_20_10, io_copy3, io_copy4,
io_inter2, io_inter3, io_pattern1, io_pattern2, io_pattern3,
io_pattern4, io_pattern5, io_pattern6, io_pattern7,
io_pattern8, io_recur1, io_recur2, io_recur3, io_scan1,
io_scan2 (io_scan2x), io_scan3, io_scan4, io_scan5, io_scan6,
io_scan7, io_scan8, io_scan13, io_scan14, io_scan15,
io_scan16, io_scan17, io_scan18, io_unif1, io_unif2,
io_unif3, io_unif4, io_unif5, io_unif6

IO.sequential 14.2
Primary : io17, io18, io19, io20, io21, io22, io23, io_copy1, io_copy2,
io_inter1, io_scan11, io_scan12
Secondary : io_inter2, io_inter3

IO.Text_IO 14.3
Primary : async1, async2, cio1, cio2, cio3, cio4, cio5, cio6, cio7,
cio8, cio9, cio10, cio11, cio12, cio13, cio14,
inst1 (inst2..inst5), io0, io1, io2, io3, io4, io5, io6, io7,
io8, io9, io10, io24, io25, io26, io27, io28, io29, io30
Secondary : async4, async5, claim23
Incidental : ss537, ss538, ss539, ss540

IO.Text_IO.float_string 14.3.8
Primary : ss134, ss135, ss136

IO.Text_IO.integer_string 14.3.7
Primary : ss137, ss431

loop.exit 5.7
Primary : ss354, ss355, ss356, ss357
Secondary : ss182, ss183, ss184, ss250, ss376, ss377, ss386, ss612
Incidental : ss406, ss427

```

loop.for                                     5.5
    Primary      : ss58, ss104, ss105, ss181, ss422, ss424, ss516, ss517, ss518,
                  ss519, ss520, ss535, ss542x
    Secondary    : claim09, claim11, ss57, ss80, ss81, ss106, ss171, ss180,
                  ss225, ss236, ss237, ss238, ss239, ss240, ss387, ss409,
                  ss423, ss425, ss525, ss536, ss541, ss542, ss651, ss749,
                  ss750, ss752, ss776
    Incidental   : ss120, ss163, ss164, ss165, ss166, ss167, ss212, ss213,
                  ss428, ss431, ss438, ss439, ss440, ss441, ss442, ss443,
                  ss472, ss473, ss477, ss490, ss491, ss511, ss512, ss654,
                  ss655, ss659, ss660, ss664, ss665, ss669, ss670, ss674,
                  ss675, ss679, ss680, ss686x, ss686y, ss689, ss690, ss694,
                  ss695, ss699, ss700, ss704, ss705, ss709, ss710, ss714,
                  ss715, ss741

loop.while                                   5.5
    Primary      : ss209, ss426
    Secondary    : ss185
    Incidental   : ss148, ss162, ss165, ss166, ss369, ss385, ss479, ss480,
                  ss481, ss482

math_dep.adx                                 4.5
    Primary      : ss810
    Secondary    : ss807

math_dep.intexp                              4.5
    Primary      : ss809
    Secondary    : ss806

math_dep.setexp                              4.5
    Primary      : ss811
    Secondary    : ss808

math.function.arcsin                         4.5
    Primary      : ss586

math.function.arctan                         4.5
    Primary      : ss34, ss299
    Incidental   : kalman, whet1, whet2, whet3, whet4

math.function.cos                            4.5
    Primary      : ss28, ss295
    Incidental   : kalman, whet1, whet2, whet3, whet4

math.function.exp                            4.5
    Primary      : ss14, ss31, ss296
    Incidental   : ss308, whet1, whet2, whet3, whet4

```

math.function.log	4.5
Primary	: ss32, ss297
Secondary	: ss14
Incidental	: ss308, whet1, whet2, whet3, whet4
math.function.sgn	4.5
Primary	: ss35
Incidental	: ss267, ss268, ss269, ss413, ss414, ss562
math.function.sin	4.5
Primary	: ss27, ss294
Incidental	: kalman, whet1, whet2, whet3, whet4
math.function.sqrt	4.5
Primary	: ss33, ss298
Incidental	: kalman, whet1, whet2, whet3, whet4
optimization.algebraic_simplification	10.6
Primary	: ss44, ss47, ss48, ss49, ss50, ss51, ss61, ss62, ss63, ss64, ss65, ss66, ss67, ss73, ss74, ss218, ss220, ss221, ss319, ss320, ss321, ss322, ss432 (ss433), ss434, ss435, ss436, ss437, ss560 (ss561)
optimization.boolean_var_elim	10.6
Primary	: ss176 (ss177)
optimization.bounds_check	10.6
Primary	: ss174, ss192, ss193, ss194, ss368
optimization.common_sub_expr_elim	10.6
Primary	: alias2, alias6 (alias6x), alias10, alias14, common, cse1, cse2, cse3, cse4, cse5, cse6, cse7, cse8, cse9, cse10, pure1 (pure2), pure5 (pure6), ss75, ss76, ss172, ss210 (ss211), ss406, ss428, ss508, ss509, ss530, ss533, ss553, ss554, ss643, ss644
optimization.constant_propagation	11.6
Primary	: firth6 (firth6x), ss316, ss317, ss529
optimization.data_flow	10.6
Primary	: ss504, ss505, ss753 (ss757), ss754 (ss757), ss755 (ss757), ss756 (ss757)
optimization.dead	10.6
Primary	: alias3, alias7 (alias7x), alias11, alias15, dead, ss56, ss68, ss71, ss225, ss226, ss427, ss638, ss639, ss640, ss641, ss642, ss649, ss650, ss651

```

optimization.folding                                10.6
    Primary    : alias4, alias8 (alias8x), alias12, alias16, fold1, fold2,
                fold3, fold4, fold5, fold6, fold7, fold8, fold_mod, ss41,
                ss41_mod, ss42, ss42_mod, ss55, ss60, ss70 (ss69), ss185,
                ss189 (ss190), ss216, ss216_mod, ss217, ss219, ss219_mod,
                ss227, ss230, ss231, ss232, ss239, ss285, ss303, ss304,
                ss305, ss306, ss314 (ss315), ss318, ss325, ss362, ss421,
                ss532, ss537, ss538, ss539, ss540, ss556, ss558 (ss559),
                ss561x, ss563, ss564, ss565, ss587 (ss591..ss594),
                ss588 (ss591..ss594), ss589 (ss591..ss594),
                ss590 (ss591..ss594), ss595, ss806, ss807, ss808
    Secondary   : ss2, ss8, ss54, ss83
optimization.inline                                10.6
    Primary    : ss260, ss410 (ss411)
optimization.jump_tracing                          10.6
    Primary    : ss182, ss183, ss184, ss250, ss619, ss620
optimization.loop_flattening                        10.6
    Primary    : ss405
optimization.loop_fusion                            10.6
    Primary    : ss180 (ss181)
optimization.loop_induction                         10.6
    Primary    : ss236, ss237, ss409
optimization.loop_interchange                       10.6
    Primary    : ss750
optimization.loop_invariant                         10.6
    Primary    : alias1, alias5 (alias5x), alias9, alias13, invar,
                pure3 (pure4), pure7 (pure8), ss212, ss222, ss429, ss430,
                ss536, ss749, ss752
optimization.loop_rotation                          10.6
    Primary    : ss385 (ss385x), ss386, ss387
optimization.loop_unrolling                         10.6
    Primary    : ss238, ss240, ss541, ss542 (ss542x)
    Secondary   : ss105
optimization.machine_idiom                          10.6
    Primary    : idioms, ss40, ss43, ss45, ss52, ss59, ss173, ss196, ss197,
                ss198, ss199, ss200, ss204, ss205 (ss206), ss207, ss208,
                ss214, ss215, ss323, ss385x, ss407, ss408, ss503, ss555,
                ss611
    Secondary   : ss7, ss29, ss30, ss115

```

optimization.merge_tests 10.6
 Primary : ss175, ss178 (ss179), ss440 (ss441)
 optimization.order_of_evaluation 10.6
 Primary : ss413, ss414, ss415, ss416, ss417, ss418, ss545, ss546,
 ss547, ss548, ss549, ss550, ss551, ss552
 optimization.redundant_code 10.6
 Primary : ss195, ss261, ss376, ss377
 Secondary : ss93
 optimization.register_allocation 10.6
 Primary : ss235, ss262, ss263, ss264, ss265, ss307, ss388, ss412,
 ss442, ss443, ss507, ss510, ss511, ss512, ss531, ss534,
 ss557, ss606, ss607, ss608, ss609, ss610, ss612
 optimization.strength_reduction 10.6
 Primary : ss15, ss16, ss188, ss213 (ss422), ss279, ss291, ss423 (ss424)
 ss425, strength
 Secondary : ss426
 optimization.test_swapping 10.6
 Primary : ss438, ss439
 optimization.unreachable_code 10.6
 Primary : ss543, ss751, unreach
 package.overhead 8.
 Primary : d_library_1, d_library_2, d_library_3, d_library_5,
 d_library_6, d_library_7, d_library_8, s_library_1,
 s_library_2, s_library_3, s_library_5, s_library_6,
 s_library_7, s_library_8, ss469, ss470, ss471, ss472, ss473,
 ss474, ss475, ss476, ss477, ss779, ss780, ss781, ss782,
 ss783, ss784, ss785, ss786, ss787, ss788
 parameters 6.4.1
 Primary : ss419 (ss420), ss584, ss585
 parameters.default 6.4.2
 Primary : ss124, ss125, ss126
 parameters.modes 6.2
 Primary : ss138, ss139, ss140, ss145, ss146, ss147, ss378, ss562
 parameters.passing 6.4
 Primary : ss566, ss567, ss568, ss569, ss570, ss571, ss572, ss573,
 ss574, ss575, ss576, ss577, ss578, ss579, ss580, ss581,
 ss582, ss583
 Secondary : ss247, ss248, ss249, ss613, ss614, ss615, ss616, ss617,
 ss618

pragma.numeric_error 11.1
 Primary : ss444, ss445, ss446, ss447, ss448, ss449, ss450, ss451
 pragma.pack 13.1
 Primary : ss156, ss157, ss158, ss159, ss160, ss161
 pragma.suppress.discriminant_check 11.7
 Primary : ss613, ss614, ss615, ss616, ss617, ss618
 Secondary : ss242
 pragma.suppress.elaboration_check 11.7
 Primary : elab1, elab10, elab2, elab3, elab4, elab5, elab6, elab7,
 elab8, elab9
 pragma.suppress.index_check 11.7
 Primary : ss53, ss54
 pragma.suppress.range_check 11.7
 Primary : firth7 (firth7x), ss117, ss168, ss169, ss170, ss171, ss363,
 ss364, ss365, ss366, ss367, ss372, ss373, ss374, ss375,
 ss757
 Secondary : ss242, ss252, ss254, ss255, ss758, ss759, ss760, ss761,
 ss762, ss763
 record.aggregates 4.3.1
 Primary : firth3 (firth3x), ss116
 record.assignment 3.7.4
 Primary : firth1 (firth1x), ss100, ss114
 record.component.assignment 3.7
 Primary : ss21, ss115, ss244
 Secondary : ss156, ss157, ss158, ss159, ss160, ss161, ss215, ss724_mod,
 ss725_mod, ss736_mod, ss737_mod, ss738_mod
 Incidental : ss407
 record.discriminants 3.7.1
 Primary : ss152, ss153, ss242, ss245, ss598 (ss599), ss600 (ss601),
 ss602, ss603, ss604, ss605
 record.operations 3.7.4
 Primary : complex_record01, complex_record02, complex_record03,
 complex_record04, complex_record05, complex_record06,
 complex_record07, complex_record08, complex_record09,
 firth2 (firth2y), firth2x, io_mem1, io_mem2, io_mem3, slice1,
 slice2, slice3, slice4, slice5, slice6, slice7, slice8,
 ss513, ss514, ss515
 record.overhead 3.7.4
 Primary : ss789, ss790, ss791, ss792, ss793, ss794, ss795, ss796,
 ss797, ss798

representation.attributes	13.7.2
Primary	: ss730_mod, ss731_mod, ss732_mod, ss734_mod, ss735_mod, ss736_mod, ss737_mod, ss738_mod, ss739_mod, ss740_mod
representation.pack.unpack	13.1
Primary	: ss652, ss653, ss654, ss655, ss656, ss657, ss658, ss659, ss660, ss661, ss662, ss663, ss664, ss665, ss666, ss667, ss668, ss669, ss670, ss671, ss672, ss673, ss674, ss675, ss676, ss677, ss678, ss679, ss680, ss681, ss687, ss688, ss689, ss690, ss691, ss692, ss693, ss694, ss695, ss696, ss697, ss698, ss699, ss700, ss701, ss702, ss703, ss704, ss705, ss706, ss707, ss708, ss709, ss710, ss711, ss712, ss713, ss714, ss715, ss716, ss724_mod, ss725_mod
scope.intermediate	8.3
Primary	: ss96_mod, ss97_mod, ss98_mod
scope.local	8.2
Primary	: ss20, ss95_mod
statement.block	5.6
Primary	: ss22, ss23, ss24, ss25, ss544
Secondary	: claim10, claim12, claim13, claim16, claim38
statement.case	5.4
Primary	: ss118, ss119
Secondary	: ss133, ss325
Incidental	: ss482, ss488
statement.goto	5.9
Primary	: ss26
Secondary	: kernel16_goto, ss261, ss385x, ss619, ss620
Incidental	: claim09, claim10, claim11, ss356
statement.if.coding_style	5.3
Primary	: firth4 (firth4x), ss82, ss83, ss84, ss85, ss86, ss87, ss88, ss89, ss90, ss91, ss92, ss94, ss186, ss187, ss223, ss224, ss490, ss491, ss494 (ss495), ss496, ss497, ss498 (ss499)

statement.if.condition 5.3

Primary : ss93, ss129, ss144, ss179, ss206, ss292, ss441, ss559

Secondary : consistent1, consistent2, consistent3, consistent4,
consistent5, consistent6, consistent7, ss207, ss208, ss324,
ss328, ss421, ss438, ss439, ss440, ss504, ss505, ss507,
ss508, ss509, ss510, ss558, ss561, ss644, ss649, ss650,
ss686x, ss686y, ss751, ss800, ss801, ss802

Incidental : ss128, ss132, ss176, ss177, ss178, ss205, ss214, ss227,
ss228, ss229, ss230, ss231, ss232, ss262, ss263, ss264,
ss311, ss312, ss313, ss319, ss320, ss321, ss322, ss339,
ss355, ss356, ss385x, ss398, ss399, ss400, ss402, ss403,
ss404, ss409, ss417, ss418, ss431, ss479, ss480, ss481,
ss511, ss512, ss526, ss527, ss528, ss537, ss538, ss539,
ss540, ss754

statement.null 5.1

Primary : label, ss0, ss106, ss804

Secondary : ss544

Incidental : ss543

statement.overhead 5.2

Primary : ss634, ss635, ss636, ss637

storage.reclamation 4.8

Primary : claim01, claim02, claim03, claim04, claim05, claim06,
claim07, claim08, claim09, claim10, claim11, claim12,
claim13, claim14, claim15, claim16, claim17, claim18,
claim19, claim20, claim21, claim22, claim23, claim24,
claim25, claim26, claim27, claim28, claim29, claim30,
claim31, claim32, claim33, claim34, claim35, claim36,
claim37, claim38, claim39, claim40, claim41, claim42,
claim43, claim44, claim45, claim46, claim47,
reclaim_collection_constrained,
reclaim_collection_unconstrained,
reclaim_global_heap_constrained,
reclaim_global_heap_unconstrained, ss162, ss163, ss164,
ss165, ss166, ss167, ss741

```

subprogram.external                                6.4
    Primary    : activation1, firth5 (firth5v..firth5z), ss36, ss37, ss38,
                ss39, ss632
    Secondary  : ss641, ss642
    Incidental : ss236, ss237, ss365, ss385, ss386, ss387, ss516, ss517,
                ss518, ss519, ss520, ss546, ss547, ss548, ss549, ss596,
                ss638, ss639, ss640, ss730_mod

subprogram.inline                                6.3.2
    Primary    : activation2, ss142 (ss144), ss411, ss633
    Secondary  : claim40, claim41, claim42, claim43, claim44, claim45,
                claim46, claim47, ss563, ss564, ss565

subprogram.local                                6.4
    Primary    : ss127, ss141, ss143, ss247, ss248, ss249, ss258, ss358,
                ss359, ss360, ss370, ss483, ss484, ss485, ss521, ss522,
                ss523, tak
    Secondary  : claim01, claim02, claim03, claim04, claim05, claim06,
                claim07, claim08, claim14, claim15, claim24, ss260, ss596,
                ss748
    Incidental : ss236, ss237, ss379, ss380, ss381, ss382, ss383, ss384,
                ss486, ss487, ss492, ss598, ss599, ss600, ss601, ss603,
                ss604, ss605

subprogram.nested                                8.3
    Primary    : ss361

task.interrupt                                    13.5.1
    Primary    : int_0, int_1, int_2, int_3, int_4, int_5, int_6, int_7,
                int_8, int_9

task.language_feature_tests                        9.
    Primary    : async3, async4, async5, delay_abort1, delay_abort2, task1,
                task2, task3, task4, task5, task6, task11, task12, task13,
                task14, task15, task16, task17, task18, task19, task20,
                task21, task22, task23, task24, task26, task27, task28,
                task29, task30, task31, task32, task33, task34, task34_delta,
                task35, task35_delta, task36, task37a, task37b, task38,
                task39, task40, task41, task42, task43, task44a, task44b,
                task45a, task45b, task46, task46x, task47, task48, task49,
                task50, task51, task52, task53, task57, task58, task59,
                task60
    Secondary  : claim34, claim35, claim36, claim37, ss740_mod

```

task.rendezvous 9.5
 Primary : task_num_1, task_num_5, task_num_10, task_num_15,
 task_num_20, task_num_25, task_num_30, task2_num_1,
 task2_num_5, task2_num_10, task2_num_15, task2_num_20,
 task2_num_25, task2_num_30
 Secondary : claim28, claim29, claim30, claim31, claim32, claim33

task.storage_size 9.9
 Primary : task54_mod, task55_mod, task56

timing.calendar 9.6
 Primary : ss453, ss454, ss456, ss457, ss799, ss800, ss801, ss802,
 ss803

timing.clock 9.6
 Primary : ss452
 Secondary : claim39

type.character.operations 3.5.5
 Primary : ss479, ss480, ss481, ss482, ss493
 Incidental : ss486, ss487, ss488, ss489, ss492

type.enumeration.attributes 3.5.5
 Primary : ss128 (ss129), ss130, ss251, ss252, ss253, ss254, ss255

type.enumeration.operations 3.5.5
 Primary : ss132, ss133, ss309, ss310

type.named_number 3.2
 Primary : ss267 (ss268..ss269), ss726_mod, ss727_mod, ss728_mod
 Secondary : ss483, ss484, ss587
 Incidental : ss529, ss530, ss531, ss534

type.string.assignment 3.6.3
 Primary : ss99, ss111, ss112, ss151, ss243, ss371
 Secondary : ss113, ss149, ss370

withdrawn.tests
 Primary : ai_create_delete_kb, ai_create_object, ai_load_kb_from_file,
 ai_modify_object, ai_query, dhrys1, dhrys2, dhrys3, fold,
 queens, ss95, ss96, ss97, ss98, ss686, ss724, ss725, ss726,
 ss727, ss728, ss729, ss730, ss731, ss732, ss734, ss735,
 ss736, ss737, ss738, ss739, ss740, task54, task55

5.6 Appendix VI, ACEC KEYWORD INDEX - 2

This appendix contains a list of test problems with their primary purposes (which may be for comparison with other tests).

a_star	Primary	: application.ai
acker1	Primary	: classical.ackermann's
acker2	Primary	: classical.ackermann's
activation1	Primary	: subprogram.external
activation2	Primary	: subprogram.inline
ai_create_delete_kb	Primary	: withdrawn.tests
ai_create_object	Primary	: withdrawn.tests
ai_load_kb_from_file	Primary	: withdrawn.tests
ai_modify_object	Primary	: withdrawn.tests
ai_query	Primary	: withdrawn.tests
alias1	Primary	: optimization.loop_invariant
alias2	Primary	: optimization.common_sub_expr_elim
alias3	Primary	: optimization.dead
alias4	Primary	: optimization.folding
alias5	Primary	: optimization.loop_invariant
alias5x	Comparison	: alias5
alias6	Primary	: optimization.common_sub_expr_elim
alias6x	Comparison	: alias6
alias7	Primary	: optimization.dead
alias7x	Comparison	: alias7
alias8	Primary	: optimization.folding
alias8x	Comparison	: alias8
alias9	Primary	: optimization.loop_invariant
alias10	Primary	: optimization.common_sub_expr_elim
alias11	Primary	: optimization.dead
alias12	Primary	: optimization.folding
alias13	Primary	: optimization.loop_invariant
alias14	Primary	: optimization.common_sub_expr_elim
alias15	Primary	: optimization.dead
alias16	Primary	: optimization.folding
arti_asum	Primary	: application.avionics
arti_atan2	Primary	: application.avionics
arti_cos	Primary	: application.avionics
arti_fmod	Primary	: application.avionics
arti_ifpm_control	Primary	: application.avionics
arti_ifpm_init	Primary	: application.avionics
arti_ifpm_io	Primary	: application.avionics
arti_ifpm_rotors	Primary	: application.avionics
arti_nairini	Primary	: application.avionics

arti_nscni	Primary	: application.avionics
arti_nutmini	Primary	: application.avionics
arti_sin	Primary	: application.avionics
async1	Primary	: IO.Text_IO
async2	Primary	: IO.Text_IO
async3	Primary	: task.language_feature_tests
	Secondary	: delay.problems
async4	Primary	: task.language_feature_tests
	Secondary	: IO.Text_IO
async5	Primary	: task.language_feature_tests
	Secondary	: IO.Text_IO
auto	Primary	: classical.numerical.comp_fam_arch(CFA)
avl_0	Primary	: application.avl_tree
avl_1	Primary	: application.avl_tree
avl_2	Primary	: application.avl_tree
avl_3	Primary	: application.avl_tree
avl_4	Primary	: application.avl_tree
avl_5	Primary	: application.avl_tree
avl_6	Primary	: application.avl_tree
avl_7	Primary	: application.avl_tree
avl_8	Primary	: application.avl_tree
avl_9	Primary	: application.avl_tree
avl_10	Primary	: application.avl_tree
avl_11	Primary	: application.avl_tree
bmt	Primary	: classical.numerical.comp_fam_arch(CFA)
bsort1	Primary	: classical.sort
bsort2	Primary	: classical.sort
cat1	Primary	: array.operations
cat2	Primary	: array.operations
cat3	Primary	: array.operations
	Secondary	: exception.raise
	Incidental	: exception.handling
cio1	Primary	: IO.Text_IO
cio2	Primary	: IO.Text_IO
cio3	Primary	: IO.Text_IO
cio4	Primary	: IO.Text_IO
cio5	Primary	: IO.Text_IO
cio6	Primary	: IO.Text_IO
cio7	Primary	: IO.Text_IO
cio8	Primary	: IO.Text_IO

cio9	Primary	: IO.Text_IO
cio10	Primary	: IO.Text_IO
cio11	Primary	: IO.Text_IO
cio12	Primary	: IO.Text_IO
cio13	Primary	: IO.Text_IO
cio14	Primary	: IO.Text_IO
ciqsort	Primary	: classical.sort
claim01	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim02	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim03	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim04	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim05	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim06	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim07	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim08	Primary	: storage.reclamation
	Secondary	: subprogram.local
claim09	Primary	: storage.reclamation
	Secondary	: loop.for
	Incidental	: statement.goto
claim10	Primary	: storage.reclamation
	Secondary	: statement.block
	Incidental	: statement.goto
claim11	Primary	: storage.reclamation
	Secondary	: loop.for
	Incidental	: statement.goto
claim12	Primary	: storage.reclamation
	Secondary	: statement.block
	Incidental	: exception.handling
claim13	Primary	: storage.reclamation
	Secondary	: exception.raise
		statement.block
	Incidental	: exception.handling

claim14	Primary : storage.reclamation
	Secondary : subprogram.local
	Incidental : exception.handling
claim15	Primary : storage.reclamation
	Secondary : subprogram.local
	Incidental : exception.handling
claim16	Primary : storage.reclamation
	Secondary : statement.block
claim17	Primary : storage.reclamation
	Secondary : image
claim18	Primary : storage.reclamation
	Secondary : array.operations
claim19	Primary : storage.reclamation
	Secondary : array.operations
	Incidental : exception.handling
claim20	Primary : storage.reclamation
	Secondary : array.operations
claim21	Primary : storage.reclamation
	Secondary : array.operations
	Incidental : exception.handling
claim22	Primary : storage.reclamation
	Secondary : array.operations
	Incidental : exception.handling
claim23	Primary : storage.reclamation
	Secondary : IO.Text_IO
claim24	Primary : storage.reclamation
	Secondary : subprogram.local
claim25	Primary : storage.reclamation
claim26	Primary : storage.reclamation
claim27	Primary : storage.reclamation
claim28	Primary : storage.reclamation
	Secondary : task.rendezvous
claim29	Primary : storage.reclamation
	Secondary : task.rendezvous
claim30	Primary : storage.reclamation
	Secondary : task.rendezvous
claim31	Primary : storage.reclamation
	Secondary : task.rendezvous
claim32	Primary : storage.reclamation
	Secondary : task.rendezvous

claim33	Primary	: storage.reclamation
	Secondary	: task.rendezvous
claim34	Primary	: storage.reclamation
	Secondary	: task.language_feature_tests
claim35	Primary	: storage.reclamation
	Secondary	: task.language_feature_tests
claim36	Primary	: storage.reclamation
	Secondary	: task.language_feature_tests
claim37	Primary	: storage.reclamation
	Secondary	: task.language_feature_tests
claim38	Primary	: storage.reclamation
	Secondary	: statement.block
	Incidental	: exception.handling
claim39	Primary	: storage.reclamation
	Secondary	: timing.clock
claim40	Primary	: storage.reclamation
	Secondary	: subprogram.inline
claim41	Primary	: storage.reclamation
	Secondary	: subprogram.inline
claim42	Primary	: storage.reclamation
	Secondary	: subprogram.inline
claim43	Primary	: storage.reclamation
	Secondary	: subprogram.inline
claim44	Primary	: storage.reclamation
	Secondary	: subprogram.inline
claim45	Primary	: storage.reclamation
	Secondary	: subprogram.inline
claim46	Primary	: storage.reclamation
	Secondary	: subprogram.inline
	Incidental	: exception.handling
claim47	Primary	: storage.reclamation
	Secondary	: subprogram.inline
	Incidental	: exception.handling
common	Primary	: optimization.common_sub_expr_elim
complex_record01	Primary	: record.operations
complex_record02	Primary	: record.operations
complex_record03	Primary	: record.operations
complex_record04	Primary	: record.operations
complex_record05	Primary	: record.operations
complex_record06	Primary	: record.operations

complex_record07	Primary	: record.operations
complex_record08	Primary	: record.operations
complex_record09	Primary	: record.operations
consistent1	Primary	: consistency_check.timing_loop
	Secondary	: statement.if.condition
consistent2	Primary	: consistency_check.timing_loop
	Secondary	: statement.if.condition
consistent3	Primary	: consistency_check.timing_loop
	Secondary	: statement.if.condition
consistent4	Primary	: consistency_check.timing_loop
	Secondary	: statement.if.condition
consistent5	Primary	: consistency_check.timing_loop
	Secondary	: statement.if.condition
consistent6	Primary	: consistency_check.timing_loop
	Secondary	: statement.if.condition
consistent7	Primary	: consistency_check.timing_loop
	Secondary	: statement.if.condition
crc0	Primary	: application.cyclic_redundancy_check
crc1	Primary	: application.cyclic_redundancy_check
crc2	Primary	: application.cyclic_redundancy_check
crc3	Primary	: application.cyclic_redundancy_check
crc4	Primary	: application.cyclic_redundancy_check
cse1	Primary	: optimization.common_sub_expr_elim
cse2	Primary	: optimization.common_sub_expr_elim
cse3	Primary	: optimization.common_sub_expr_elim
cse4	Primary	: optimization.common_sub_expr_elim
cse5	Primary	: optimization.common_sub_expr_elim
cse6	Primary	: optimization.common_sub_expr_elim
cse7	Primary	: optimization.common_sub_expr_elim
cse8	Primary	: optimization.common_sub_expr_elim
cse9	Primary	: optimization.common_sub_expr_elim
cse10	Primary	: optimization.common_sub_expr_elim
d_library_1	Primary	: package.overhead
d_library_2	Primary	: package.overhead
d_library_3	Primary	: package.overhead
d_library_5	Primary	: package.overhead
d_library_6	Primary	: package.overhead
d_library_7	Primary	: package.overhead
d_library_8	Primary	: package.overhead
dead	Primary	: optimization.dead

delay1	Primary	: delay.problems
delay2	Primary	: delay.problems
delay3	Primary	: delay.problems
delay4	Primary	: delay.problems
delay5	Primary	: delay.problems
delay6	Primary	: delay.problems
delay7	Primary	: delay.problems
delay8	Primary	: delay.problems
delay9	Primary	: delay.problems
delay10	Primary	: delay.problems
delay11	Primary	: delay.problems
delay12	Primary	: delay.problems
delay13	Primary	: delay.problems
delay14	Primary	: delay.problems
delay_abort1	Primary	: task.language_feature_tests
	Secondary	: delay.problems
delay_abort2	Primary	: task.language_feature_tests
	Secondary	: delay.problems
delay_zero0	Primary	: delay.problems
delay_zero1	Primary	: delay.problems
delay_zero2	Primary	: delay.problems
delay_zero3	Primary	: delay.problems
delay_zero4	Primary	: delay.problems
delay_zero5	Primary	: delay.problems
delay_zero6	Primary	: delay.problems
delay_zero6x	Comparison	: delay_zero6
delay_zero7	Primary	: delay.problems
delay_zero8	Primary	: delay.problems
des1	Primary	: application.data_encryption_standard
des2	Primary	: application.data_encryption_standard
des3	Primary	: application.data_encryption_standard
des4	Primary	: application.data_encryption_standard
des4a	Primary	: application.data_encryption_standard
des5	Primary	: application.data_encryption_standard
des5a	Primary	: application.data_encryption_standard
des6	Primary	: application.data_encryption_standard
des6a	Primary	: application.data_encryption_standard
des7	Primary	: application.data_encryption_standard
des7a	Primary	: application.data_encryption_standard

dhrys1	Primary	: withdrawn.tests
dhrys1_mod	Primary	: classical.dhrystone
dhrys2	Primary	: withdrawn.tests
dhrys2_mod	Primary	: classical.dhrystone
dhrys3	Primary	: withdrawn.tests
dhrys3_mod	Primary	: classical.dhrystone
elab1	Primary	: pragma.suppress.elaboration_check
elab10	Primary	: pragma.suppress.elaboration_check
elab2	Primary	: pragma.suppress.elaboration_check
elab3	Primary	: pragma.suppress.elaboration_check
elab4	Primary	: pragma.suppress.elaboration_check
elab5	Primary	: pragma.suppress.elaboration_check
elab6	Primary	: pragma.suppress.elaboration_check
elab7	Primary	: pragma.suppress.elaboration_check
elab8	Primary	: pragma.suppress.elaboration_check
elab9	Primary	: pragma.suppress.elaboration_check
enum_io1	Primary	: generic.instanstiation
enum_io2	Primary	: generic.instanstiation
enum_io3	Primary	: generic.instanstiation
enum_io4	Primary	: generic.instanstiation
enum_io5	Primary	: generic.instanstiation
enum_io6	Primary	: generic.instanstiation
enum_io7	Primary	: generic.instanstiation
enum_io8	Primary	: generic.instanstiation
enum_io9	Primary	: generic.instanstiation
ew	Primary	: application.avionics
filter1	Primary	: application.filter
	Secondary	: generic.subprogram
filter1i	Primary	: application.filter
	Secondary	: generic.subprogram
filter2	Primary	: application.filter
	Secondary	: generic.package
filter2i	Primary	: application.filter
	Secondary	: generic.package
filter3	Primary	: application.filter
filter4	Primary	: application.filter
firth1	Primary	: record.assignment
firth1x	Comparison	: firth1

firth2	Primary : record.operations
firth2x	Primary : record.operations
firth2y	Comparison : firth2
firth3	Primary : record.aggregates
firth3x	Comparison : firth3
firth4	Primary : statement.if.coding_style
firth4x	Comparison : firth4
firth5	Primary : subprogram.external
firth5v	Comparison : firth5
firth5w	Comparison : firth5
firth5x	Comparison : firth5
firth5y	Comparison : firth5
firth5z	Comparison : firth5
firth6	Primary : optimization.constant_propagation
firth6x	Comparison : firth6
firth7	Primary : pragma.suppress.range_check
firth7x	Comparison : firth7
fold	Primary : withdrawn.tests
fold1	Primary : optimization.folding
fold2	Primary : optimization.folding
fold3	Primary : optimization.folding
fold4	Primary : optimization.folding
fold5	Primary : optimization.folding
fold6	Primary : optimization.folding
fold7	Primary : optimization.folding
fold8	Primary : optimization.folding
fold_mod	Primary : optimization.folding
forward_euler1	Primary : application.avionics
forward_euler2	Primary : application.avionics
funcexcp	Primary : exception.handling
gamm	Primary : classical.GAMM_measure
gamm2	Primary : classical.GAMM_measure
heapify	Primary : classical.numerical.comp_fam_arch(CFA)
idioms	Primary : optimization.machine_idiom
inst1	Primary : IO.Text_IO
inst2	Comparison : inst1
inst3	Comparison : inst1
inst4	Comparison : inst1
	Incidental : image
inst5	Comparison : inst1

int_0	Primary	: task.interrupt
int_1	Primary	: task.interrupt
int_2	Primary	: task.interrupt
int_3	Primary	: task.interrupt
int_4	Primary	: task.interrupt
int_5	Primary	: task.interrupt
int_6	Primary	: task.interrupt
int_7	Primary	: task.interrupt
int_8	Primary	: task.interrupt
int_9	Primary	: task.interrupt
invar	Primary	: optimization.loop_invariant
io0	Primary	: IO.Text_IO
io1	Primary	: IO.Text_IO
io2	Primary	: IO.Text_IO
io3	Primary	: IO.Text_IO
io4	Primary	: IO.Text_IO
io5	Primary	: IO.Text_IO
io6	Primary	: IO.Text_IO
io7	Primary	: IO.Text_IO
io8	Primary	: IO.Text_IO
io9	Primary	: IO.Text_IO
io10	Primary	: IO.Text_IO
io11	Primary	: IO.direct
io12	Primary	: IO.direct
io13	Primary	: IO.direct
io14	Primary	: IO.direct
io15	Primary	: IO.direct
io16	Primary	: IO.direct
io17	Primary	: IO.sequential
io18	Primary	: IO.sequential
io19	Primary	: IO.sequential
io20	Primary	: IO.sequential
io21	Primary	: IO.sequential
io22	Primary	: IO.sequential
io23	Primary	: IO.sequential
io24	Primary	: IO.Text_IO
io25	Primary	: IO.Text_IO
io26	Primary	: IO.Text_IO
io27	Primary	: IO.Text_IO
io28	Primary	: IO.Text_IO

io29	Primary	: IO.Text_IO
io30	Primary	: IO.Text_IO
io_80_20_1	Primary	: IO.direct
io_80_20_2	Primary	: IO.direct
io_80_20_3	Primary	: IO.direct
io_80_20_4	Primary	: IO.direct
io_80_20_5	Primary	: IO.direct
io_80_20_6	Primary	: IO.direct
io_80_20_7	Primary	: IO.direct
io_80_20_8	Primary	: IO.direct
io_80_20_9	Primary	: IO.direct
io_80_20_10	Primary	: IO.direct
io_copy1	Primary	: IO.sequential
io_copy2	Primary	: IO.sequential
io_copy3	Primary	: IO.direct
io_copy4	Primary	: IO.direct
io_inter1	Primary	: IO.sequential
io_inter2	Primary	: IO.direct
	Secondary	: IO.sequential
io_inter3	Primary	: IO.direct
	Secondary	: IO.sequential
io_mem1	Primary	: record.operations
io_mem2	Primary	: record.operations
io_mem3	Primary	: record.operations
io_pattern1	Primary	: IO.direct
io_pattern2	Primary	: IO.direct
io_pattern3	Primary	: IO.direct
io_pattern4	Primary	: IO.direct
io_pattern5	Primary	: IO.direct
io_pattern6	Primary	: IO.direct
io_pattern7	Primary	: IO.direct
io_pattern8	Primary	: IO.direct
io_recur1	Primary	: IO.direct
io_recur2	Primary	: IO.direct
io_recur3	Primary	: IO.direct
io_scan1	Primary	: IO.direct
io_scan2	Primary	: IO.direct
io_scan2x	Comparison	: io_scan2
io_scan3	Primary	: IO.direct
io_scan4	Primary	: IO.direct

io_scan5	Primary	: IO.direct
io_scan6	Primary	: IO.direct
io_scan7	Primary	: IO.direct
io_scan8	Primary	: IO.direct
io_scan11	Primary	: IO.sequential
io_scan12	Primary	: IO.sequential
io_scan13	Primary	: IO.direct
io_scan14	Primary	: IO.direct
io_scan15	Primary	: IO.direct
io_scan16	Primary	: IO.direct
io_scan17	Primary	: IO.direct
io_scan18	Primary	: IO.direct
io_unif1	Primary	: IO.direct
io_unif2	Primary	: IO.direct
io_unif3	Primary	: IO.direct
io_unif4	Primary	: IO.direct
io_unif5	Primary	: IO.direct
io_unif6	Primary	: IO.direct
iqsort	Primary	: classical.sort
kalman	Primary	: application.kalman_filter
	Incidental	: math.function.arctan
		math.function.cos
		math.function.sin
		math.function.sqrt
kernel1	Primary	: classical.numerical.livermore_loops
kernel2	Primary	: classical.numerical.livermore_loops
kernel3	Primary	: classical.numerical.livermore_loops
kernel4	Primary	: classical.numerical.livermore_loops
kernel5	Primary	: classical.numerical.livermore_loops
kernel6	Primary	: classical.numerical.livermore_loops
kernel7	Primary	: classical.numerical.livermore_loops
kernel8	Primary	: classical.numerical.livermore_loops
kernel9	Primary	: classical.numerical.livermore_loops
kernel10	Primary	: classical.numerical.livermore_loops
kernel11	Primary	: classical.numerical.livermore_loops
kernel12	Primary	: classical.numerical.livermore_loops
kernel13	Primary	: classical.numerical.livermore_loops
kernel14	Primary	: classical.numerical.livermore_loops
kernel15	Primary	: classical.numerical.livermore_loops

kernel16	Primary	: classical.numerical.livermore_loops
kernel16_goto	Comparison	: kernel16
	Secondary	: statement.goto
kernel17	Primary	: classical.numerical.livermore_loops
kernel18	Primary	: classical.numerical.livermore_loops
kernel19	Primary	: classical.numerical.livermore_loops
kernel20	Primary	: classical.numerical.livermore_loops
kernel21	Primary	: classical.numerical.livermore_loops
kernel22	Primary	: classical.numerical.livermore_loops
kernel23	Primary	: classical.numerical.livermore_loops
kernel24	Primary	: classical.numerical.livermore_loops
label	Primary	: statement.null
loop0	Primary	: classical.numerical.knuth_loops
loop1	Primary	: classical.numerical.knuth_loops
loop2	Primary	: classical.numerical.knuth_loops
loop3	Primary	: classical.numerical.knuth_loops
loop4a	Primary	: classical.numerical.knuth_loops
loop4b	Primary	: classical.numerical.knuth_loops
loop4c	Primary	: classical.numerical.knuth_loops
loop5	Primary	: classical.numerical.knuth_loops
loop6	Primary	: classical.numerical.knuth_loops
loop7	Primary	: classical.numerical.knuth_loops
loop8	Primary	: classical.numerical.knuth_loops
loop9	Primary	: classical.numerical.knuth_loops
loop10	Primary	: classical.numerical.knuth_loops
loop11	Primary	: classical.numerical.knuth_loops
loop12	Primary	: classical.numerical.knuth_loops
loop13	Primary	: classical.numerical.knuth_loops
loop14	Primary	: classical.numerical.knuth_loops
loop15	Primary	: classical.numerical.knuth_loops
loop16	Primary	: classical.numerical.knuth_loops
loop17	Primary	: classical.numerical.knuth_loops
lu	Primary	: classical.numerical.comp_fam_arch(CFA)
merge1	Primary	: classical.sort
merge2	Primary	: classical.sort
neural	Primary	: application.ai
pure1	Primary	: optimization.common_sub_expr_elim
pure2	Comparison	: pure1
pure3	Primary	: optimization.loop_invariant
pure4	Comparison	: pure3

pure5	Primary	: optimization.common_sub_expr_elim
pure6	Comparison	: pure5
pure7	Primary	: optimization.loop_invariant
pure8	Comparison	: pure7
puzzle	Primary	: classical.cube_placing
qsort1	Primary	: classical.sort
qsort2	Primary	: classical.sort
queens	Primary	: withdrawn.tests
queens_mod	Primary	: classical.eight_queens
reclaim_collection_constrained	Primary	: storage.reclamation
	Secondary	: access.operations
reclaim_collection_unconstrained	Primary	: storage.reclamation
	Secondary	: access.operations
reclaim_global_heap_constrained	Primary	: storage.reclamation
	Secondary	: access.operations
reclaim_global_heap_unconstrained	Primary	: storage.reclamation
	Secondary	: access.operations
reed_solomon_0	Primary	: application.error_correcting_code
reed_solomon_1	Primary	: application.error_correcting_code
reed_solomon_2	Primary	: application.error_correcting_code
reed_solomon_3	Primary	: application.error_correcting_code
reed_solomon_4	Primary	: application.error_correcting_code
runge	Primary	: classical.numerical.comp_fam_arch(CFA)
s_library_1	Primary	: package.overhead
s_library_2	Primary	: package.overhead
s_library_3	Primary	: package.overhead
s_library_5	Primary	: package.overhead
s_library_6	Primary	: package.overhead
s_library_7	Primary	: package.overhead
s_library_8	Primary	: package.overhead
search	Primary	: classical.search
shell1	Primary	: classical.sort
shell2	Primary	: classical.sort
sieve	Primary	: classical.prime_number

simulate_bmbat	Primary	: application.simulation
simulate_emrpm	Primary	: application.simulation
simulate_hmproto	Primary	: application.simulation
simulate_qmpitch	Primary	: application.simulation
simulate_rcwfrdet	Primary	: application.simulation
simulate_umnav	Primary	: application.simulation
simulate_kmdump	Primary	: application.simulation
simulate_rmkeying	Primary	: application.simulation
slice1	Primary	: record.operations
slice2	Primary	: record.operations
slice3	Primary	: record.operations
slice4	Primary	: record.operations
slice5	Primary	: record.operations
slice6	Primary	: record.operations
slice7	Primary	: record.operations
slice8	Primary	: record.operations
ss0	Primary	: statement.null
ss1	Primary	: float.operations
ss2	Primary	: conversion.float
	Secondary	: optimization.folding
ss2_mod1	Primary	: conversion.float
ss2_mod2	Comparison	: ss2_mod1
ss3	Primary	: float.operations
ss4	Primary	: float.operations
ss5	Primary	: float.operations
ss6	Primary	: float.operations
ss7	Primary	: integer.operations
	Secondary	: optimization.machine_idiom
ss8	Primary	: conversion.integer
	Secondary	: optimization.folding
ss8_mod	Primary	: conversion.integer
ss9	Primary	: integer.operations
ss10	Primary	: integer.operations
ss11	Primary	: integer.operations
ss12	Primary	: conversion.integer
ss13	Primary	: conversion.float
ss14	Primary	: math.function.exp
	Secondary	: math.function.log
ss15	Primary	: optimization.strength_reduction
	Secondary	: expression.exponentiating

ss16	Primary	: optimization.strength_reduction
	Secondary	: expression.exponentiating
ss17	Primary	: array.operations
ss18	Primary	: array.operations
ss19	Primary	: array.operations
ss20	Primary	: scope.local
	Secondary	: float.operations
ss21	Primary	: record.component.assignment
	Secondary	: expression.exponentiating float.operations
ss22	Primary	: statement.block
	Secondary	: float.operations
ss23	Primary	: statement.block
	Secondary	: float.operations
ss24	Primary	: statement.block
	Secondary	: float.operations
ss25	Primary	: statement.block
	Secondary	: float.operations
ss26	Primary	: statement.goto
ss27	Primary	: math.function.sin
ss28	Primary	: math.function.cos
ss29	Primary	: expression.abs
	Secondary	: optimization.machine_idiom
ss30	Primary	: expression.abs
	Secondary	: optimization.machine_idiom
ss31	Primary	: math.function.exp
ss32	Primary	: math.function.log
ss33	Primary	: math.function.sqrt
ss34	Primary	: math.function.arctan
ss35	Primary	: math.function.sgn
ss36	Primary	: subprogram.external
ss37	Primary	: subprogram.external
ss38	Primary	: subprogram.external
ss39	Primary	: subprogram.external
ss40	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss41	Primary	: optimization.folding
	Secondary	: integer.operations
ss41_mod	Primary	: optimization.folding
	Secondary	: integer.operations

ss42	Primary	: optimization.folding
	Secondary	: integer.operations
ss42_mod	Primary	: optimization.folding
	Secondary	: integer.operations
ss43	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss44	Primary	: optimization.algebraic_simplification
	Secondary	: integer.operations
ss45	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss46	Primary	: integer.operations
ss47	Primary	: optimization.algebraic_simplification
	Secondary	: integer.operations
ss48	Primary	: optimization.algebraic_simplification
	Secondary	: integer.operations
ss49	Primary	: optimization.algebraic_simplification
	Secondary	: integer.operations
ss50	Primary	: optimization.algebraic_simplification
	Secondary	: expression.exponentiating integer.operations
ss51	Primary	: optimization.algebraic_simplification
	Secondary	: expression.exponentiating integer.operations
ss52	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss53	Primary	: pragma.suppress.index_check
	Secondary	: array.operations
ss54	Primary	: pragma.suppress.index_check
	Secondary	: array.operations optimization.folding
ss55	Primary	: optimization.folding
	Secondary	: array.operations
ss56	Primary	: optimization.dead
	Secondary	: integer.operations
ss57	Primary	: array.operations
	Secondary	: loop.for
ss58	Primary	: loop.for
	Secondary	: array.operations
ss59	Primary	: optimization.machine_idiom
	Secondary	: float.operations

ss60	Primary	: optimization.folding
	Secondary	: float.operations
ss61	Primary	: optimization.algebraic_simplification
	Secondary	: float.operations
ss62	Primary	: optimization.algebraic_simplification
	Secondary	: float.operations
ss63	Primary	: optimization.algebraic_simplification
	Secondary	: float.operations
ss64	Primary	: optimization.algebraic_simplification
	Secondary	: float.operations
ss65	Primary	: optimization.algebraic_simplification
	Secondary	: expression.exponentiating float.operations
ss66	Primary	: optimization.algebraic_simplification
	Secondary	: expression.exponentiating float.operations
ss67	Primary	: optimization.algebraic_simplification
	Incidental	: float.operations
ss68	Primary	: optimization.dead
	Incidental	: float.operations
ss69	Comparison	: ss70
	Incidental	: float.operations
ss70	Primary	: optimization.folding
	Incidental	: float.operations
ss71	Primary	: optimization.dead
	Secondary	: float.operations
ss72	Primary	: boolean.expressions
ss73	Primary	: optimization.algebraic_simplification
	Secondary	: boolean.expressions
ss74	Primary	: optimization.algebraic_simplification
	Secondary	: boolean.expressions
ss75	Primary	: optimization.common_sub_expr_elim
	Secondary	: array.operations
ss76	Primary	: optimization.common_sub_expr_elim
	Secondary	: array.operations
ss77	Primary	: array.operations
ss78	Primary	: array.operations
ss79	Primary	: array.operations
ss80	Primary	: array.operations
	Secondary	: loop.for

ss81	Primary	: array.operations
	Secondary	: loop.for
ss82	Primary	: statement.if.coding_style
ss83	Primary	: statement.if.coding_style
	Secondary	: optimization.folding
ss84	Primary	: statement.if.coding_style
ss85	Primary	: statement.if.coding_style
ss86	Primary	: statement.if.coding_style
ss87	Primary	: statement.if.coding_style
ss88	Primary	: statement.if.coding_style
ss89	Primary	: statement.if.coding_style
ss90	Primary	: statement.if.coding_style
ss91	Primary	: statement.if.coding_style
ss92	Primary	: statement.if.coding_style
ss93	Primary	: statement.if.condition
	Secondary	: optimization.redundant_code
ss94	Primary	: statement.if.coding_style
ss95	Primary	: withdrawn.tests
ss95_mod	Primary	: scope.local
	Incidental	: integer.operations
ss96	Primary	: withdrawn.tests
ss96_mod	Primary	: scope.intermediate
	Incidental	: integer.operations
ss97	Primary	: withdrawn.tests
ss97_mod	Primary	: scope.intermediate
	Incidental	: integer.operations
ss98	Primary	: withdrawn.tests
ss98_mod	Primary	: scope.intermediate
	Incidental	: integer.operations
ss99	Primary	: type.string.assignment
ss100	Primary	: record.assignment
ss101	Primary	: boolean.expressions
ss102	Primary	: integer.MOD
	Incidental	: integer.operations
ss103	Primary	: integer.REM
	Incidental	: integer.operations
ss104	Primary	: loop.for
ss105	Primary	: loop.for
	Secondary	: optimization.loop_unrolling

ss106	Primary : statement.null
	Secondary : loop.for
ss107	Primary : conversion.fixed
ss108	Primary : conversion.fixed
ss109	Primary : fixed.operations
ss110	Primary : fixed.operations
ss111	Primary : type.string.assignment
ss112	Primary : type.string.assignment
ss113	Primary : expression.catenation
	Secondary : type.string.assignment
ss114	Primary : record.assignment
ss115	Primary : record.component.assignment
	Secondary : optimization.machine_idiom
ss116	Primary : record.aggregates
ss117	Primary : exception.raise
	pragma.suppress.range_check
	Incidental : integer.operations
ss118	Primary : statement.case
ss119	Primary : statement.case
ss120	Primary : application.polynomial.coding_style
	Secondary : array.operations
	Incidental : float.operations
	loop.for
ss121	Primary : application.polynomial.coding_style
	Incidental : float.operations
ss122	Primary : application.polynomial.coding_style
	Incidental : float.operations
ss123	Primary : application.polynomial.coding_style
	Incidental : float.operations
ss124	Primary : parameters.default
ss125	Primary : parameters.default
ss126	Primary : parameters.default
ss127	Primary : subprogram.local
ss128	Primary : type.enumeration.attributes
	Incidental : statement.if.condition
ss129	Primary : statement.if.condition
	Comparison : ss128
	Incidental : integer.operations
ss130	Primary : type.enumeration.attributes
	Incidental : integer.operations

ss131	Primary : image
	Incidental : integer.operations
ss132	Primary : type.enumeration.operations
	Incidental : statement.if.condition
ss133	Primary : type.enumeration.operations
	Secondary : statement.case
ss134	Primary : IO.Text_IO.float_string
	Secondary : float.operations
ss135	Primary : IO.Text_IO.float_string
	Secondary : float.operations
ss136	Primary : IO.Text_IO.float_string
	Secondary : float.operations
ss137	Primary : IO.Text_IO.integer_string
	Secondary : integer.operations
ss138	Primary : parameters.modes
	Incidental : integer.operations
ss139	Primary : parameters.modes
	Incidental : integer.operations
ss140	Primary : parameters.modes
	Incidental : integer.operations
ss141	Primary : subprogram.local
	Incidental : float.operations
ss142	Primary : subprogram.inline
	Incidental : float.operations
ss143	Primary : subprogram.local
	Incidental : float.operations
ss144	Primary : statement.if.condition
	Comparison : ss142
ss145	Primary : parameters.modes
	Incidental : boolean.expressions
ss146	Primary : parameters.modes
	Incidental : boolean.expressions
ss147	Primary : parameters.modes
	Incidental : boolean.expressions
ss148	Primary : generic.subprogram
	Incidental : loop.while
ss149	Primary : generic.subprogram
	Secondary : type.string.assignment
ss150	Primary : generic.subprogram
	Secondary : float.operations

ss151	Primary : type.string.assignment Comparison : ss149
ss152	Primary : record.discriminants
ss153	Primary : record.discriminants
ss154	Primary : access.operations Incidental : float.operations
ss155	Primary : access.operations Incidental : float.operations
ss156	Primary : pragma.pack Secondary : record.component.assignment
ss157	Primary : pragma.pack Secondary : record.component.assignment
ss158	Primary : pragma.pack Secondary : record.component.assignment
ss159	Primary : pragma.pack Secondary : record.component.assignment
ss160	Primary : pragma.pack Secondary : record.component.assignment
ss161	Primary : pragma.pack Secondary : access.operations record.component.assignment
ss162	Primary : storage.reclamation Secondary : access.operations Incidental : loop.while
ss163	Primary : storage.reclamation Secondary : access.operations Incidental : loop.for
ss164	Primary : storage.reclamation Secondary : access.operations Incidental : loop.for
ss165	Primary : storage.reclamation Secondary : access.operations Incidental : loop.for loop.while
ss166	Primary : storage.reclamation Secondary : access.operations Incidental : loop.for loop.while

ss167	Primary : storage.reclamation Secondary : access.operations Incidental : loop.for
ss168	Primary : pragma.suppress.range_check Secondary : array.operations
ss169	Primary : pragma.suppress.range_check Secondary : array.operations
ss170	Primary : pragma.suppress.range_check Secondary : array.operations
ss171	Primary : pragma.suppress.range_check Secondary : loop.for
ss172	Primary : optimization.common_sub_expr_elim Secondary : array.operations
ss173	Primary : optimization.machine_idiom Secondary : array.operations
ss174	Primary : optimization.bounds_check Secondary : array.operations
ss175	Primary : optimization.merge_tests Secondary : array.operations
ss176	Primary : optimization.boolean_var_elim Secondary : boolean.expressions Incidental : statement.if.condition
ss177	Primary : boolean.expressions Comparison : ss176 Incidental : statement.if.condition
ss178	Primary : optimization.merge_tests Incidental : statement.if.condition
ss179	Primary : statement.if.condition Comparison : ss178
ss180	Primary : optimization.loop_fusion Secondary : loop.for
ss181	Primary : loop.for Comparison : ss180
ss182	Primary : optimization.jump_tracing Secondary : loop.exit
ss183	Primary : optimization.jump_tracing Secondary : loop.exit
ss184	Primary : optimization.jump_tracing Secondary : loop.exit

ss185	Primary	: optimization.folding
	Secondary	: loop.while
ss186	Primary	: statement.if.coding_style
ss187	Primary	: statement.if.coding_style
ss188	Primary	: optimization.strength_reduction
	Secondary	: expression.exponentiating
ss189	Primary	: optimization.folding
	Secondary	: integer.operations
ss190	Comparison	: ss189
	Incidental	: integer.operations
ss191	Primary	: expression.exponentiating
	Incidental	: integer.operations
ss192	Primary	: optimization.bounds_check
	Secondary	: array.operations
ss193	Primary	: optimization.bounds_check
	Secondary	: array.operations
ss194	Primary	: optimization.bounds_check
	Secondary	: array.operations
ss195	Primary	: optimization.redundant_code
	Secondary	: integer.operations
ss196	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss197	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss198	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss199	Primary	: optimization.machine_idiom
	Secondary	: integer.MOD
ss200	Primary	: optimization.machine_idiom
	Incidental	: integer.operations
ss201	Primary	: integer.operations
ss202	Primary	: integer.operations
ss203	Primary	: integer.operations
ss204	Primary	: optimization.machine_idiom
	Secondary	: integer.REM
ss205	Primary	: optimization.machine_idiom
	Incidental	: statement.if.condition
ss206	Primary	: statement.if.condition
	Comparison	: ss205

ss207	Primary : optimization.machine_idiom
	Secondary : statement.if.condition
ss208	Primary : optimization.machine_idiom
	Secondary : statement.if.condition
ss209	Primary : loop.while
	Incidental : integer.operations
ss210	Primary : optimization.common_sub_expr_elim
	Incidental : float.operations
ss211	Primary : float.operations
	Comparison : ss210
ss212	Primary : optimization.loop_invariant
	Incidental : loop.for
ss213	Primary : optimization.strength_reduction
	Secondary : expression.exponentiating
	Incidental : integer.operations
	loop.for
ss214	Primary : optimization.machine_idiom
	Incidental : integer.operations
	statement.if.condition
ss215	Primary : optimization.machine_idiom
	Secondary : record.component.assignment
ss216	Primary : optimization.folding
	Secondary : expression.exponentiating
	float.operations
ss216_mod	Primary : optimization.folding
	Secondary : expression.exponentiating
	float.operations
ss217	Primary : optimization.folding
	Secondary : expression.exponentiating
	integer.operations
ss218	Primary : optimization.algebraic_simplification
	Incidental : float.operations
ss219	Primary : optimization.folding
	Secondary : expression.exponentiating
	float.operations
ss219_mod	Primary : optimization.folding
	Secondary : expression.exponentiating
	float.operations
ss220	Primary : optimization.algebraic_simplification
	Secondary : float.operations

ss221	Primary : optimization.algebraic_simplification Secondary : integer.operations
ss222	Primary : optimization.loop_invariant
ss223	Primary : statement.if.coding_style
ss224	Primary : statement.if.coding_style
ss225	Primary : optimization.dead Secondary : loop.for
ss226	Primary : optimization.dead Incidental : float.operations
ss227	Primary : optimization.folding Secondary : boolean.expressions Incidental : statement.if.condition
ss228	Primary : boolean.expressions Incidental : statement.if.condition
ss229	Primary : boolean.expressions Incidental : statement.if.condition
ss230	Primary : optimization.folding Secondary : boolean.expressions Incidental : statement.if.condition
ss231	Primary : optimization.folding Secondary : boolean.expressions Incidental : statement.if.condition
ss232	Primary : optimization.folding Secondary : boolean.expressions Incidental : statement.if.condition
ss233	Primary : conversion.integer Incidental : float.operations
ss234	Primary : conversion.integer Incidental : float.operations
ss235	Primary : optimization.register_allocation Secondary : array.operations
ss236	Primary : optimization.loop_induction Secondary : loop.for Incidental : subprogram.external subprogram.local
ss237	Primary : optimization.loop_induction Secondary : loop.for Incidental : subprogram.external subprogram.local

ss238	Primary	: optimization.loop_unrolling
	Secondary	: loop.for
ss239	Primary	: optimization.folding
	Secondary	: loop.for
ss240	Primary	: optimization.loop_unrolling
	Secondary	: loop.for
ss241	Primary	: conversion.null
	Incidental	: integer.operations
ss242	Primary	: record.discriminants
	Secondary	: pragma.suppress.discriminant_check pragma.suppress.range_check
ss243	Primary	: type.string.assignment
	Secondary	: array.operations
ss244	Primary	: record.component.assignment
ss245	Primary	: record.discriminants
ss246	Primary	: expression.attributes
	Secondary	: array.operations
ss247	Primary	: subprogram.local
	Secondary	: parameters.passing
ss248	Primary	: subprogram.local
	Secondary	: parameters.passing
ss249	Primary	: subprogram.local
	Secondary	: parameters.passing
ss250	Primary	: optimization.jump_tracing
	Secondary	: loop.exit
ss251	Primary	: type.enumeration.attributes
ss252	Primary	: type.enumeration.attributes
	Secondary	: pragma.suppress.range_check
ss253	Primary	: type.enumeration.attributes
ss254	Primary	: type.enumeration.attributes
	Secondary	: pragma.suppress.range_check
ss255	Primary	: type.enumeration.attributes
	Secondary	: pragma.suppress.range_check
ss256	Primary	: access.operations
	Secondary	: float.operations
ss257	Primary	: access.operations
	Secondary	: float.operations
ss258	Primary	: subprogram.local
	Comparison	: ss259
	Secondary	: array.operations

ss259	Primary : conversion.unchecked_conversion
	Secondary : array.operations
ss260	Primary : optimization.inline
	Secondary : subprogram.local
ss261	Primary : optimization.redundant_code
	Secondary : statement.goto
ss262	Primary : optimization.register_allocation
	Incidental : float.operations
	statement.if.condition
ss263	Primary : optimization.register_allocation
	Incidental : float.operations
	statement.if.condition
ss264	Primary : optimization.register_allocation
	Incidental : integer.operations
	statement.if.condition
ss265	Primary : optimization.register_allocation
	Incidental : integer.operations
ss266	Primary : expression.abs
	Incidental : integer.operations
ss267	Primary : type.named_number
	Incidental : integer.operations
	math.function.sgn
ss268	Primary : integer.operations
	Comparison : ss267
	Incidental : math.function.sgn
ss269	Primary : integer.operations
	Comparison : ss267
	Incidental : math.function.sgn
ss270	Primary : integer.bigint.operations
ss271	Primary : integer.bigint.operations
ss272	Primary : integer.bigint.operations
ss273	Primary : integer.bigint.operations
ss274	Primary : integer.bigint.operations
ss275	Primary : integer.bigint.operations
ss276	Primary : integer.bigint.operations
	Incidental : integer.REM
ss277	Primary : integer.bigint.operations
	Secondary : conversion.integer
ss278	Primary : integer.bigint.operations

ss279	Primary	: optimization.strength_reduction
	Secondary	: expression.exponentiating
ss280	Primary	: integer.bigint.operations
	Secondary	: boolean.expressions
ss281	Primary	: integer.operations
ss282	Primary	: integer.bigint.operations
	Secondary	: conversion.integer
ss283	Primary	: integer.bigint.operations
	Secondary	: conversion.float
ss284	Primary	: integer.bigint.operations
	Secondary	: array.operations
ss285	Primary	: optimization.folding
	Secondary	: array.operations
ss286	Primary	: float.operations
ss287	Primary	: float.operations
ss288	Primary	: float.operations
ss289	Primary	: conversion.float
ss290	Primary	: conversion.float
ss291	Primary	: optimization.strength_reduction
	Secondary	: expression.exponentiating
	Incidental	: float.operations
ss292	Primary	: statement.if.condition
	Incidental	: float.operations
ss293	Primary	: expression.abs
	Secondary	: float.operations
ss294	Primary	: math.function.sin
	Secondary	: float.operations
ss295	Primary	: math.function.cos
	Secondary	: float.operations
ss296	Primary	: math.function.exp
	Secondary	: float.operations
ss297	Primary	: math.function.log
	Secondary	: float.operations
ss298	Primary	: math.function.sqrt
	Secondary	: float.operations
ss299	Primary	: math.function.arctan
	Secondary	: float.operations
ss300	Primary	: conversion.integer
ss301	Primary	: array.operations
	Secondary	: float.operations

ss302	Primary	: float.operations
ss303	Primary	: optimization.folding
	Secondary	: conversion.integer
ss304	Primary	: optimization.folding
	Secondary	: expression.exponentiating
	Incidental	: float.operations
ss305	Primary	: optimization.folding
	Secondary	: expression.exponentiating
	Incidental	: float.operations
ss306	Primary	: optimization.folding
	Secondary	: expression.exponentiating
	Incidental	: float.operations
ss307	Primary	: optimization.register_allocation
	Secondary	: expression.exponentiating
	Incidental	: float.operations
ss308	Primary	: float.operations
	Incidental	: math.function.exp math.function.log
ss309	Primary	: type.enumeration.operations
	Secondary	: array.operations
ss310	Primary	: type.enumeration.operations
ss311	Primary	: exception.raise
	Incidental	: statement.if.condition
ss312	Primary	: exception.raise
	Incidental	: statement.if.condition
ss313	Primary	: exception.numeric_error
	Incidental	: statement.if.condition
ss314	Primary	: optimization.folding
	Secondary	: float.operations
	Incidental	: boolean.expressions
ss315	Primary	: float.operations
	Comparison	: ss314
	Incidental	: boolean.expressions
ss316	Primary	: optimization.constant_propagation
	Secondary	: float.operations
	Incidental	: boolean.expressions
ss317	Primary	: optimization.constant_propagation
	Secondary	: float.operations
	Incidental	: boolean.expressions

ss318	Primary	: optimization.folding
	Secondary	: float.operations
	Incidental	: boolean.expressions
ss319	Primary	: optimization.algebraic_simplification
	Incidental	: statement.if.condition
ss320	Primary	: optimization.algebraic_simplification
	Incidental	: statement.if.condition
ss321	Primary	: optimization.algebraic_simplification
	Incidental	: statement.if.condition
ss322	Primary	: optimization.algebraic_simplification
	Incidental	: statement.if.condition
ss323	Primary	: optimization.machine_idiom
	Secondary	: float.operations
	Incidental	: boolean.expressions
ss324	Primary	: float.operations
	Secondary	: statement.if.condition
ss325	Primary	: optimization.folding
	Secondary	: statement.case
ss326	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss327	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss328	Primary	: boolean.arrays.unpacked
	Secondary	: statement.if.condition
ss329	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss330	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss331	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss332	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss333	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss334	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss335	Primary	: conversion.packed_to_unpacked
	Secondary	: boolean.expressions
ss336	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions

ss337	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss338	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss339	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
	Incidental	: statement.if.condition
ss340	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss341	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss342	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss343	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss344	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss345	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss346	Primary	: conversion.packed_to_unpacked
	Secondary	: boolean.expressions
	Incidental	: boolean.arrays.packed boolean.arrays.unpacked
ss347	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss348	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss349	Primary	: boolean.arrays.packed
	Secondary	: boolean.expressions
ss350	Primary	: conversion.unpacked_to_packed
	Secondary	: boolean.expressions
ss351	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss352	Primary	: boolean.arrays.unpacked
	Secondary	: boolean.expressions
ss353	Primary	: conversion.packed_to_unpacked
	Secondary	: boolean.expressions
	Incidental	: boolean.arrays.packed boolean.arrays.unpacked
ss354	Primary	: loop.exit

ss355	Primary : loop.exit
	Incidental : statement.if.condition
ss356	Primary : loop.exit
	Incidental : statement.goto
	statement.if.condition
ss357	Primary : loop.exit
ss358	Primary : subprogram.local
ss359	Primary : subprogram.local
ss360	Primary : subprogram.local
ss361	Primary : subprogram.nested
ss362	Primary : optimization.folding
	Incidental : integer.REM
ss363	Primary : pragma.suppress.range_check
	Incidental : integer.REM
ss364	Primary : pragma.suppress.range_check
	Incidental : integer.operations
ss365	Primary : pragma.suppress.range_check
	Incidental : subprogram.external
ss366	Primary : pragma.suppress.range_check
	Incidental : integer.operations
ss367	Primary : pragma.suppress.range_check
	Incidental : integer.operations
ss368	Primary : optimization.bounds_check
	Secondary : expression.abs
ss369	Primary : exception.numeric_error
	Incidental : integer.operations
	loop.while
ss370	Primary : subprogram.local
	Secondary : type.string.assignment
	Incidental : image
ss371	Primary : type.string.assignment
ss372	Primary : pragma.suppress.range_check
	Incidental : integer.operations
ss373	Primary : pragma.suppress.range_check
	Incidental : integer.operations
ss374	Primary : pragma.suppress.range_check
	Incidental : integer.operations
ss375	Primary : pragma.suppress.range_check
	Incidental : integer.operations

ss376	Primary : optimization.redundant_code Secondary : loop.exit
ss377	Primary : optimization.redundant_code Secondary : loop.exit
ss378	Primary : parameters.modes
ss379	Primary : exception.handling Incidental : subprogram.local
ss380	Primary : exception.handling Incidental : subprogram.local
ss381	Primary : exception.handling Incidental : subprogram.local
ss382	Primary : exception.handling Incidental : subprogram.local
ss383	Primary : exception.handling Incidental : subprogram.local
ss384	Primary : exception.handling Incidental : integer.operations subprogram.local
ss385	Primary : optimization.loop_rotation Incidental : loop.while subprogram.external
ss385x	Primary : optimization.machine_idiom Comparison : ss385 Secondary : statement.goto Incidental : integer.operations statement.if.condition
ss386	Primary : optimization.loop_rotation Secondary : loop.exit Incidental : integer.operations subprogram.external
ss387	Primary : optimization.loop_rotation Secondary : loop.for Incidental : subprogram.external
ss388	Primary : optimization.register_allocation Secondary : array.operations
ss389	Primary : expression.parenthesis Secondary : float.operations
ss390	Primary : expression.parenthesis Secondary : float.operations

ss391	Primary : expression.parenthesis
	Secondary : float.operations
ss392	Primary : expression.parenthesis
	Secondary : float.operations
ss393	Primary : expression.parenthesis
	Secondary : integer.operations
ss394	Primary : expression.parenthesis
	Secondary : integer.operations
ss395	Primary : expression.parenthesis
	Secondary : integer.operations
ss396	Primary : expression.parenthesis
	Secondary : integer.operations
ss397	Primary : application.lag_filter
	Incidental : float.operations
ss398	Primary : application.integration
	Incidental : float.operations
	statement.if.condition
ss399	Primary : application.symmetric_deadzone
	Incidental : float.operations
	statement.if.condition
ss400	Primary : application.symmetric_limiter
	Incidental : float.operations
	statement.if.condition
ss401	Primary : application.lag_filter
	Incidental : float.operations
ss402	Primary : application.integration
	Incidental : float.operations
	statement.if.condition
ss403	Primary : application.symmetric_deadzone
	Incidental : float.operations
	statement.if.condition
ss404	Primary : application.symmetric_limiter
	Incidental : float.operations
	statement.if.condition
ss405	Primary : optimization.loop_flattening
	Incidental : array.operations
ss406	Primary : optimization.common_sub_expr_elim
	Incidental : array.operations
	float.operations
	loop.exit

ss407	Primary : optimization.machine_idiom Incidental : float.operations record.component.assignment
ss408	Primary : optimization.machine_idiom
ss409	Primary : optimization.loop_induction Secondary : loop.for Incidental : array.operations statement.if.condition
ss410	Primary : optimization.inline Incidental : array.operations
ss411	Primary : subprogram.inline Comparison : ss410 Incidental : array.operations
ss412	Primary : optimization.register_allocation
ss413	Primary : optimization.order_of_evaluation Incidental : float.operations math.function.sgn
ss414	Primary : optimization.order_of_evaluation Incidental : float.operations math.function.sgn
ss415	Primary : optimization.order_of_evaluation Incidental : float.operations
ss416	Primary : optimization.order_of_evaluation Incidental : float.operations
ss417	Primary : optimization.order_of_evaluation Incidental : float.operations statement.if.condition
ss418	Primary : optimization.order_of_evaluation Incidental : float.operations statement.if.condition
ss419	Primary : array.dynamic parameters Incidental : array.operations
ss420	Comparison : ss419 Incidental : array.operations
ss421	Primary : optimization.folding Secondary : statement.if.condition
ss422	Primary : loop.for Comparison : ss213

ss423	Primary : optimization.strength_reduction Secondary : loop.for Incidental : integer.operations
ss424	Primary : loop.for Comparison : ss423 Incidental : integer.operations
ss425	Primary : optimization.strength_reduction Secondary : loop.for Incidental : integer.operations
ss426	Primary : loop.while Secondary : optimization.strength_reduction Incidental : integer.operations
ss427	Primary : optimization.dead Incidental : integer.operations loop.exit
ss428	Primary : optimization.common_sub_expr_elim Incidental : array.operations integer.operations loop.for
ss429	Primary : optimization.loop_invariant Secondary : array.operations Incidental : integer.operations
ss430	Primary : optimization.loop_invariant Secondary : array.operations Incidental : integer.operations
ss431	Primary : IO.Text_IO.integer_string Incidental : expression.abs float.operations integer.operations loop.for statement.if.condition
ss432	Primary : optimization.algebraic_simplification Incidental : array.operations float.operations
ss433	Comparison : ss432 Incidental : array.operations float.operations
ss434	Primary : optimization.algebraic_simplification Incidental : array.operations float.operations

ss435	Primary : optimization.algebraic_simplification Incidental : array.operations float.operations
ss436	Primary : optimization.algebraic_simplification Incidental : array.operations float.operations
ss437	Primary : optimization.algebraic_simplification Incidental : array.operations float.operations
ss438	Primary : optimization.test_swapping Secondary : statement.if.condition Incidental : array.operations loop.for
ss439	Primary : optimization.test_swapping Secondary : statement.if.condition Incidental : array.operations loop.for
ss440	Primary : optimization.merge_tests Secondary : statement.if.condition Incidental : integer.operations loop.for
ss441	Primary : statement.if.condition Comparison : ss440 Incidental : integer.operations loop.for
ss442	Primary : optimization.register_allocation Incidental : array.operations float.operations loop.for
ss443	Primary : optimization.register_allocation Incidental : array.operations float.operations loop.for
ss444	Primary : pragma.numeric_error Incidental : float.operations
ss445	Primary : pragma.numeric_error Incidental : integer.operations
ss446	Primary : pragma.numeric_error Incidental : integer.MOD integer.operations

ss447	Primary : pragma.numeric_error Incidental : integer.operations integer.REM
ss448	Primary : pragma.numeric_error Incidental : float.operations
ss449	Primary : pragma.numeric_error Incidental : integer.operations
ss450	Primary : pragma.numeric_error Incidental : float.operations
ss451	Primary : pragma.numeric_error Incidental : integer.operations
ss452	Primary : timing.clock
ss453	Primary : timing.calendar
ss454	Primary : timing.calendar Incidental : float.operations
ss455	Primary : delay.problems
ss456	Primary : timing.calendar
ss457	Primary : timing.calendar
ss458	Primary : delay.problems
ss459	Primary : delay.problems
ss460	Primary : fixed.operations
ss461	Primary : fixed.operations
ss462	Primary : fixed.operations
ss463	Primary : fixed.operations
ss464	Primary : fixed.operations Incidental : boolean.expressions
ss465	Primary : fixed.operations
ss466	Primary : conversion.fixed Incidental : integer.operations
ss467	Primary : conversion.fixed Incidental : float.operations
ss468	Primary : conversion.integer Incidental : integer.operations
ss469	Primary : package.overhead Incidental : integer.operations
ss470	Primary : package.overhead Incidental : integer.operations
ss471	Primary : package.overhead Incidental : integer.operations

ss472	Primary : package.overhead Incidental : integer.operations loop.for
ss473	Primary : package.overhead Incidental : integer.operations loop.for
ss474	Primary : package.overhead Incidental : integer.operations
ss475	Primary : package.overhead Incidental : integer.operations
ss476	Primary : package.overhead Incidental : integer.operations
ss477	Primary : package.overhead Incidental : array.operations loop.for
ss478	Primary : generic.subprogram
ss479	Primary : type.character.operations Incidental : loop.while statement.if.condition
ss480	Primary : type.character.operations Incidental : loop.while statement.if.condition
ss481	Primary : type.character.operations Incidental : loop.while statement.if.condition
ss482	Primary : type.character.operations Incidental : loop.while statement.case
ss483	Primary : subprogram.local Secondary : type.named_number
ss484	Primary : subprogram.local Secondary : type.named_number
ss485	Primary : subprogram.local Incidental : float.operations
ss486	Primary : boolean.expressions Incidental : boolean.arrays.unpacked subprogram.local type.character.operations

ss487	Primary : boolean.expressions Incidental : subprogram.local type.character.operations
ss488	Primary : boolean.expressions Incidental : statement.case type.character.operations
ss489	Primary : boolean.expressions Incidental : type.character.operations
ss490	Primary : statement.if.coding_style Incidental : integer.operations loop.for
ss491	Primary : statement.if.coding_style Incidental : integer.operations loop.for
ss492	Primary : boolean.expressions Incidental : subprogram.local type.character.operations
ss493	Primary : type.character.operations
ss494	Primary : statement.if.coding_style
ss495	Comparison : ss494
ss496	Primary : statement.if.coding_style
ss497	Primary : statement.if.coding_style
ss498	Primary : statement.if.coding_style
ss499	Primary : boolean.expressions Comparison : ss498
ss500	Primary : conversion.unchecked_conversion Secondary : boolean.expressions Incidental : boolean.arrays.packed integer.operations
ss501	Primary : conversion.unchecked_conversion Secondary : boolean.expressions Incidental : boolean.arrays.packed integer.operations
ss502	Primary : conversion.unchecked_conversion Secondary : boolean.expressions Incidental : boolean.arrays.packed integer.operations
ss503	Primary : optimization.machine_idiom Secondary : integer.operations

ss504	Primary : optimization.data_flow
	Secondary : statement.if.condition
ss505	Primary : optimization.data_flow
	Secondary : statement.if.condition
ss506	Primary : conversion.unchecked_conversion
	Incidental : boolean.arrays.packed
	integer.operations
ss507	Primary : optimization.register_allocation
	Secondary : statement.if.condition
	Incidental : integer.operations
ss508	Primary : optimization.common_sub_expr_elim
	Secondary : statement.if.condition
	Incidental : array.operations
ss509	Primary : optimization.common_sub_expr_elim
	Secondary : statement.if.condition
	Incidental : array.operations
ss510	Primary : optimization.register_allocation
	Secondary : statement.if.condition
ss511	Primary : optimization.register_allocation
	Secondary : array.operations
	Incidental : float.operations
	integer.operations
	loop.for
	statement.if.condition
ss512	Primary : optimization.register_allocation
	Secondary : array.operations
	Incidental : float.operations
	integer.operations
	loop.for
	statement.if.condition
ss513	Primary : record.operations
	Incidental : float.operations
ss514	Primary : record.operations
	Incidental : float.operations
ss515	Primary : record.operations
	Incidental : float.operations
ss516	Primary : loop.for
	Incidental : array.operations
	subprogram.external

ss517	Primary : loop.for Incidental : array.operations subprogram.external
ss518	Primary : loop.for Secondary : array.operations Incidental : subprogram.external
ss519	Primary : loop.for Secondary : array.operations Incidental : subprogram.external
ss520	Primary : loop.for Secondary : array.operations Incidental : subprogram.external
ss521	Primary : subprogram.local
ss522	Primary : subprogram.local
ss523	Primary : subprogram.local
ss524	Primary : boolean.arrays.packed
ss525	Primary : boolean.arrays.packed Secondary : loop.for
ss526	Primary : boolean.arrays.packed Incidental : statement.if.condition
ss527	Primary : exception.handling Incidental : statement.if.condition
ss528	Primary : exception.handling Incidental : statement.if.condition
ss529	Primary : optimization.constant_propagation Incidental : float.operations type.named_number
ss530	Primary : optimization.common_sub_expr_elim Incidental : float.operations type.named_number
ss531	Primary : optimization.register_allocation Incidental : float.operations type.named_number
ss532	Primary : optimization.folding Incidental : float.operations
ss533	Primary : optimization.common_sub_expr_elim Incidental : float.operations
ss534	Primary : optimization.register_allocation Incidental : float.operations type.named_number

ss535	Primary : loop.for Incidental : array.operations float.operations
ss536	Primary : optimization.loop_invariant Secondary : loop.for Incidental : array.operations float.operations
ss537	Primary : optimization.folding Incidental : IO.Text_IO statement.if.condition
ss538	Primary : optimization.folding Incidental : IO.Text_IO statement.if.condition
ss539	Primary : optimization.folding Incidental : IO.Text_IO statement.if.condition
ss540	Primary : optimization.folding Incidental : IO.Text_IO statement.if.condition
ss541	Primary : optimization.loop_unrolling Secondary : loop.for Incidental : array.operations
ss542	Primary : optimization.loop_unrolling Secondary : loop.for Incidental : array.operations
ss542x	Primary : loop.for Comparison : ss542 Incidental : array.operations
ss543	Primary : optimization.unreachable_code Secondary : exception.handling Incidental : statement.null
ss544	Primary : statement.block Secondary : statement.null
ss545	Primary : optimization.order_of_evaluation Incidental : array.operations
ss546	Primary : optimization.order_of_evaluation Incidental : subprogram.external
ss547	Primary : optimization.order_of_evaluation Incidental : float.operations subprogram.external

ss548	Primary : optimization.order_of_evaluation Incidental : float.operations subprogram.external
ss549	Primary : optimization.order_of_evaluation Incidental : float.operations subprogram.external
ss550	Primary : optimization.order_of_evaluation Secondary : integer.operations
ss551	Primary : optimization.order_of_evaluation Secondary : integer.operations
ss552	Primary : optimization.order_of_evaluation Secondary : float.operations
ss553	Primary : optimization.common_sub_expr_elim Secondary : array.operations
ss554	Primary : optimization.common_sub_expr_elim Secondary : array.operations
ss555	Primary : optimization.machine_idiom
ss556	Primary : optimization.folding Secondary : integer.operations
ss557	Primary : optimization.register_allocation Incidental : array.operations
ss558	Primary : optimization.folding Secondary : statement.if.condition Incidental : integer.operations
ss559	Primary : statement.if.condition Comparison : ss558 Incidental : integer.operations
ss560	Primary : optimization.algebraic_simplification Secondary : integer.operations
ss561	Primary : integer.operations Comparison : ss560 Secondary : statement.if.condition
ss561x	Primary : optimization.folding
ss562	Primary : parameters.modes Incidental : array.operations math.function.sgn
ss563	Primary : optimization.folding Secondary : subprogram.inline Incidental : integer.operations

ss564	Primary : optimization.folding
	Secondary : subprogram.inline
	Incidental : integer.operations
ss565	Primary : optimization.folding
	Secondary : subprogram.inline
	Incidental : integer.operations
ss566	Primary : parameters.passing
	Secondary : integer.operations
ss567	Primary : parameters.passing
	Secondary : integer.operations
ss568	Primary : parameters.passing
	Secondary : integer.operations
ss569	Primary : parameters.passing
	Secondary : integer.operations
ss570	Primary : parameters.passing
	Secondary : integer.operations
ss571	Primary : parameters.passing
	Secondary : integer.operations
ss572	Primary : parameters.passing
	Secondary : integer.operations
ss573	Primary : parameters.passing
	Secondary : integer.operations
ss574	Primary : parameters.passing
	Secondary : integer.operations
ss575	Primary : parameters.passing
	Secondary : float.operations
ss576	Primary : parameters.passing
	Secondary : float.operations
ss577	Primary : parameters.passing
	Secondary : float.operations
ss578	Primary : parameters.passing
	Secondary : float.operations
ss579	Primary : parameters.passing
	Secondary : float.operations
ss580	Primary : parameters.passing
	Secondary : float.operations
ss581	Primary : parameters.passing
	Secondary : float.operations
ss582	Primary : parameters.passing
	Secondary : float.operations

ss583	Primary : parameters.passing
	Secondary : float.operations
ss584	Primary : parameters
	Secondary : integer.operations
ss585	Primary : parameters
	Secondary : float.operations
ss586	Primary : math.function.arcsin
	Incidental : float.operations
ss587	Primary : optimization.folding
	Secondary : type.named_number
ss588	Primary : optimization.folding
	Secondary : float.operations
ss589	Primary : optimization.folding
	Secondary : float.operations
ss590	Primary : optimization.folding
	Secondary : float.operations
ss591	Primary : float.operations
	Comparison : ss587
	ss588
	ss589
	ss590
ss592	Comparison : ss587
	ss588
	ss589
	ss590
	ss591
ss593	Comparison : ss587
	ss588
	ss589
	ss590
	ss591
ss594	Comparison : ss587
	ss588
	ss589
	ss590
	ss591
ss595	Primary : optimization.folding
	Secondary : float.operations

ss596	Primary : array.constraints Secondary : subprogram.local Incidental : array.operations subprogram.external
ss597	Comparison : ss596 Incidental : array.operations
ss598	Primary : record.discriminants Incidental : boolean.expressions exception.handling subprogram.local
ss599	Comparison : ss598 Incidental : boolean.expressions exception.handling subprogram.local
ss600	Primary : record.discriminants Incidental : subprogram.local
ss601	Comparison : ss600 Incidental : subprogram.local
ss602	Primary : record.discriminants Incidental : boolean.expressions exception.handling
ss603	Primary : record.discriminants Incidental : subprogram.local
ss604	Primary : record.discriminants Incidental : boolean.expressions exception.handling subprogram.local
ss605	Primary : record.discriminants Incidental : subprogram.local
ss606	Primary : optimization.register_allocation Secondary : float.operations
ss607	Primary : optimization.register_allocation Secondary : float.operations
ss608	Primary : optimization.register_allocation Secondary : integer.operations
ss609	Primary : optimization.register_allocation Secondary : float.operations
ss610	Primary : optimization.register_allocation Secondary : integer.operations

ss611	Primary	: optimization.machine_idiom
	Secondary	: integer.operations
ss612	Primary	: optimization.register_allocation
	Secondary	: loop.exit
	Incidental	: integer.operations
ss613	Primary	: pragma.suppress.discriminant_check
	Secondary	: parameters.passing
ss614	Primary	: pragma.suppress.discriminant_check
	Secondary	: parameters.passing
ss615	Primary	: pragma.suppress.discriminant_check
	Secondary	: parameters.passing
ss616	Primary	: pragma.suppress.discriminant_check
	Secondary	: parameters.passing
ss617	Primary	: pragma.suppress.discriminant_check
	Secondary	: parameters.passing
ss618	Primary	: pragma.suppress.discriminant_check
	Secondary	: parameters.passing
ss619	Primary	: optimization.jump_tracing
	Secondary	: statement.goto
ss620	Primary	: optimization.jump_tracing
	Secondary	: statement.goto
ss621	Primary	: generic.subprogram
	Incidental	: float.operations
ss622	Primary	: generic.subprogram
	Incidental	: float.operations
ss623	Primary	: generic.subprogram
	Incidental	: float.operations
ss624	Primary	: generic.subprogram
	Incidental	: float.operations
ss625	Primary	: generic.subprogram
	Incidental	: float.operations
ss626	Primary	: generic.subprogram
	Incidental	: float.operations
ss627	Primary	: generic.subprogram
	Incidental	: float.operations
ss628	Primary	: generic.subprogram
	Incidental	: float.operations
ss629	Primary	: generic.subprogram
	Incidental	: float.operations

ss630	Primary : generic.subprogram Incidental : float.operations
ss631	Primary : generic.subprogram Incidental : float.operations
ss632	Primary : subprogram.external Incidental : float.operations
ss633	Primary : subprogram.inline Incidental : float.operations
ss634	Primary : statement.overhead Incidental : integer.operations
ss635	Primary : statement.overhead Incidental : integer.operations
ss636	Primary : statement.overhead Incidental : integer.operations
ss637	Primary : statement.overhead Incidental : integer.operations
ss638	Primary : optimization.dead Incidental : exception.handling integer.operations subprogram.external
ss639	Primary : optimization.dead Incidental : integer.operations subprogram.external
ss640	Primary : optimization.dead Incidental : integer.operations subprogram.external
ss641	Primary : optimization.dead Secondary : subprogram.external
ss642	Primary : optimization.dead Secondary : subprogram.external
ss643	Primary : optimization.common_sub_expr_elim Secondary : float.operations
ss643x	Primary : float.operations Secondary : expression.exponentiating
ss644	Primary : optimization.common_sub_expr_elim Secondary : statement.if.condition
ss645	Primary : array.operations Incidental : float.operations
ss646	Primary : array.operations Incidental : float.operations

ss647	Primary : array.operations Incidental : float.operations
ss648	Primary : access.operations Incidental : array.operations
ss649	Primary : optimization.dead Secondary : statement.if.condition Incidental : float.operations
ss650	Primary : optimization.dead Secondary : statement.if.condition Incidental : float.operations
ss651	Primary : optimization.dead Secondary : loop.for Incidental : integer.operations
ss652	Primary : representation.pack.unpack Incidental : array.operations integer.operations
ss653	Primary : representation.pack.unpack Incidental : array.operations
ss654	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss655	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss656	Primary : representation.pack.unpack Incidental : array.operations
ss657	Primary : representation.pack.unpack Incidental : array.operations
ss658	Primary : representation.pack.unpack Incidental : array.operations
ss659	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss660	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss661	Primary : representation.pack.unpack Incidental : array.operations
ss662	Primary : representation.pack.unpack Incidental : array.operations

ss663	Primary : representation.pack.unpack Incidental : array.operations
ss664	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss665	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss666	Primary : representation.pack.unpack Incidental : array.operations
ss667	Primary : representation.pack.unpack Incidental : array.operations
ss668	Primary : representation.pack.unpack Incidental : array.operations
ss669	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss670	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss671	Primary : representation.pack.unpack Incidental : array.operations
ss672	Primary : representation.pack.unpack Incidental : array.operations
ss673	Primary : representation.pack.unpack Incidental : array.operations
ss674	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss675	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss676	Primary : representation.pack.unpack Incidental : array.operations
ss677	Primary : representation.pack.unpack Incidental : array.operations
ss678	Primary : representation.pack.unpack Incidental : array.operations

ss679	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss680	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss681	Primary : representation.pack.unpack Incidental : array.operations
ss682	Primary : boolean.record
ss683	Primary : boolean.record
ss684	Primary : boolean.record
ss685	Primary : boolean.record
ss686	Primary : withdrawn.tests
ss686x	Primary : boolean.expressions Secondary : statement.if.condition Incidental : loop.for
ss686y	Primary : boolean.expressions Secondary : statement.if.condition Incidental : loop.for
ss687	Primary : representation.pack.unpack Incidental : array.operations
ss688	Primary : representation.pack.unpack Incidental : array.operations
ss689	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss690	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss691	Primary : representation.pack.unpack Incidental : array.operations
ss692	Primary : representation.pack.unpack Incidental : array.operations
ss693	Primary : representation.pack.unpack Incidental : array.operations
ss694	Primary : representation.pack.unpack Incidental : array.operations loop.for

ss695	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss696	Primary : representation.pack.unpack Incidental : array.operations
ss697	Primary : representation.pack.unpack Incidental : array.operations
ss698	Primary : representation.pack.unpack Incidental : array.operations
ss699	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss700	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss701	Primary : representation.pack.unpack Incidental : array.operations
ss702	Primary : representation.pack.unpack Incidental : array.operations
ss703	Primary : representation.pack.unpack Incidental : array.operations
ss704	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss705	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss706	Primary : representation.pack.unpack Incidental : array.operations
ss707	Primary : representation.pack.unpack Incidental : array.operations
ss708	Primary : representation.pack.unpack Incidental : array.operations
ss709	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss710	Primary : representation.pack.unpack Incidental : array.operations loop.for

ss711	Primary : representation.pack.unpack Incidental : array.operations
ss712	Primary : representation.pack.unpack Incidental : array.operations
ss713	Primary : representation.pack.unpack Incidental : array.operations
ss714	Primary : representation.pack.unpack Incidental : array.operations
ss715	Primary : representation.pack.unpack Incidental : array.operations loop.for
ss716	Primary : representation.pack.unpack Incidental : array.operations
ss717	Primary : boolean.record
ss718	Primary : boolean.record
ss719	Primary : boolean.record
ss720	Primary : boolean.record
ss721	Primary : conversion.fixed
ss722	Primary : conversion.fixed
ss723	Primary : conversion.fixed
ss724	Primary : withdrawn.tests
ss724_mod	Primary : representation.pack.unpack Secondary : record.component.assignment
ss725	Primary : withdrawn.tests
ss725_mod	Primary : representation.pack.unpack Secondary : record.component.assignment
ss726	Primary : withdrawn.tests
ss726_mod	Primary : type.named_number
ss727	Primary : withdrawn.tests
ss727_mod	Primary : type.named_number
ss728	Primary : withdrawn.tests
ss728_mod	Primary : type.named_number
ss729	Primary : withdrawn.tests
ss729_mod	Primary : integer.operations
ss730	Primary : withdrawn.tests
ss730_mod	Primary : representation.attributes Incidental : subprogram.external

ss731	Primary	: withdrawn.tests
ss731_mod	Primary	: representation.attributes
	Incidental	: array.operations
ss732	Primary	: withdrawn.tests
ss732_mod	Primary	: representation.attributes
	Incidental	: array.operations
ss734	Primary	: withdrawn.tests
ss734_mod	Primary	: representation.attributes
	Incidental	: array.operations
ss735	Primary	: withdrawn.tests
ss735_mod	Primary	: representation.attributes
	Incidental	: array.operations
ss736	Primary	: withdrawn.tests
ss736_mod	Primary	: representation.attributes
	Secondary	: record.component.assignment
ss737	Primary	: withdrawn.tests
ss737_mod	Primary	: representation.attributes
	Secondary	: record.component.assignment
ss738	Primary	: withdrawn.tests
ss738_mod	Primary	: representation.attributes
	Secondary	: record.component.assignment
ss739	Primary	: withdrawn.tests
ss739_mod	Primary	: representation.attributes
	Secondary	: access.operations
ss740	Primary	: withdrawn.tests
ss740_mod	Primary	: representation.attributes
	Secondary	: task.language_feature_tests
ss741	Primary	: storage.reclamation
	Incidental	: exception.handling
		loop.for
ss744	Primary	: integer.operations
	Comparison	: ss746
ss745	Primary	: integer.operations
	Comparison	: ss746
ss746	Primary	: access.operations
ss747	Primary	: interface.language.assembly
ss748	Primary	: access.operations
	Secondary	: subprogram.local

ss749	Primary : optimization.loop_invariant
	Secondary : loop.for
	Incidental : array.operations
ss750	Primary : optimization.loop_interchange
	Secondary : loop.for
	Incidental : array.operations
ss751	Primary : optimization.unreachable_code
	Secondary : statement.if.condition
ss752	Primary : optimization.loop_invariant
	Secondary : loop.for
	Incidental : integer.operations
ss753	Primary : optimization.data_flow
	Secondary : integer.operations
	Incidental : float.operations
ss754	Primary : optimization.data_flow
	Secondary : integer.operations
	Incidental : float.operations
	statement.if.condition
ss755	Primary : optimization.data_flow
	Secondary : exception.raise
	Incidental : integer.operations
ss756	Primary : optimization.data_flow
	Incidental : integer.operations
ss757	Primary : pragma.suppress.range_check
	Comparison : ss753
	ss754
	ss755
	ss756
	Secondary : exception.raise
	Incidental : integer.operations
ss758	Primary : array.operations
	Secondary : pragma.suppress.range_check
	Incidental : float.operations
ss759	Primary : array.operations
	Secondary : pragma.suppress.range_check
	Incidental : float.operations
ss760	Primary : array.operations
	Secondary : pragma.suppress.range_check
	Incidental : float.operations

ss761	Primary : array.operations
	Secondary : pragma.suppress.range_check
	Incidental : float.operations
ss762	Primary : array.operations
	Secondary : pragma.suppress.range_check
	Incidental : float.operations
ss763	Primary : array.operations
	Secondary : pragma.suppress.range_check
	Incidental : float.operations
ss764	Primary : boolean.arrays.packed
	Secondary : array.aggregates
ss765	Primary : boolean.arrays.packed
	Secondary : array.aggregates
ss766	Primary : boolean.arrays.packed
	Secondary : array.aggregates
ss767	Primary : boolean.arrays.packed
	Secondary : array.aggregates
ss768	Primary : boolean.arrays.packed
	Secondary : array.aggregates
ss769	Primary : consistency_check.timing_loop
	Comparison : ss768
ss770	Primary : consistency_check.timing_loop
	Comparison : ss768
ss771	Primary : consistency_check.timing_loop
	Comparison : ss768
ss772	Primary : consistency_check.timing_loop
	Comparison : ss768
ss773	Primary : consistency_check.timing_loop
	Comparison : ss768
ss774	Primary : array.operations
	Incidental : integer.operations
ss775	Primary : array.aggregates
	Incidental : integer.operations
ss776	Primary : array.operations
	Secondary : loop.for
	Incidental : integer.operations
ss777	Primary : array.operations
	Incidental : integer.operations
ss778	Primary : array.aggregates
	Incidental : integer.operations

ss779	Primary	: package.overhead
	Secondary	: float.operations
ss780	Primary	: package.overhead
	Secondary	: float.operations
ss781	Primary	: package.overhead
	Secondary	: float.operations
ss782	Primary	: package.overhead
	Secondary	: float.operations
ss783	Primary	: package.overhead
	Secondary	: float.operations
ss784	Primary	: package.overhead
	Secondary	: float.operations
ss785	Primary	: package.overhead
	Secondary	: float.operations
ss786	Primary	: package.overhead
	Secondary	: float.operations
ss787	Primary	: package.overhead
	Secondary	: float.operations
ss788	Primary	: package.overhead
	Secondary	: float.operations
ss789	Primary	: record.overhead
	Secondary	: float.operations
ss790	Primary	: record.overhead
	Secondary	: float.operations
ss791	Primary	: record.overhead
	Secondary	: float.operations
ss792	Primary	: record.overhead
	Secondary	: float.operations
ss793	Primary	: record.overhead
	Secondary	: float.operations
ss794	Primary	: record.overhead
	Secondary	: float.operations
ss795	Primary	: record.overhead
	Secondary	: float.operations
ss796	Primary	: record.overhead
	Secondary	: float.operations
ss797	Primary	: record.overhead
	Secondary	: float.operations
ss798	Primary	: record.overhead
	Secondary	: float.operations

ss799	Primary	: timing.calendar
ss800	Primary	: timing.calendar
	Secondary	: statement.if.condition
ss801	Primary	: timing.calendar
	Secondary	: statement.if.condition
ss802	Primary	: timing.calendar
	Secondary	: statement.if.condition
ss803	Primary	: timing.calendar
ss804	Primary	: statement.null
ss805	Primary	: access.operations
	Incidental	: boolean.expressions
ss806	Primary	: optimization.folding
	Secondary	: generic.package math_dep.intexp
ss807	Primary	: optimization.folding
	Secondary	: generic.package math_dep.adx
ss808	Primary	: optimization.folding
	Secondary	: generic.package math_dep.setexp
ss809	Primary	: math_dep.intexp
	Secondary	: generic.package
ss810	Primary	: math_dep.adx
	Secondary	: generic.package
ss811	Primary	: math_dep.setexp
	Secondary	: generic.package
ssearch	Primary	: classical.search
ssearch2	Primary	: classical.search
strength	Primary	: optimization.strength_reduction
tak	Primary	: subprogram.local
target	Primary	: classical.numerical.comp_fam_arch(CFA)
task1	Primary	: task.language_feature_tests
task2	Primary	: task.language_feature_tests
task3	Primary	: task.language_feature_tests
task4	Primary	: task.language_feature_tests
task5	Primary	: task.language_feature_tests
task6	Primary	: task.language_feature_tests
task7	Primary	: classical.dining_philosophers
task8	Primary	: classical.dining_philosophers
task9	Primary	: classical.dining_philosophers

task10	Primary	: classical.dining_philosophers
task11	Primary	: task.language_feature_tests
task12	Primary	: task.language_feature_tests
task13	Primary	: task.language_feature_tests
task14	Primary	: task.language_feature_tests
task15	Primary	: task.language_feature_tests
task16	Primary	: task.language_feature_tests
task17	Primary	: task.language_feature_tests
task18	Primary	: task.language_feature_tests
task19	Primary	: task.language_feature_tests
task20	Primary	: task.language_feature_tests
task21	Primary	: task.language_feature_tests
task22	Primary	: task.language_feature_tests
task23	Primary	: task.language_feature_tests
task24	Primary	: task.language_feature_tests
task25	Primary	: classical.dining_philosophers
task26	Primary	: task.language_feature_tests
task27	Primary	: task.language_feature_tests
task28	Primary	: task.language_feature_tests
task29	Primary	: task.language_feature_tests
task30	Primary	: task.language_feature_tests
task31	Primary	: task.language_feature_tests
task32	Primary	: task.language_feature_tests
task33	Primary	: task.language_feature_tests
task34	Primary	: task.language_feature_tests
task34_delta	Primary	: task.language_feature_tests
task35	Primary	: task.language_feature_tests
task35_delta	Primary	: task.language_feature_tests
task36	Primary	: task.language_feature_tests
task37a	Primary	: task.language_feature_tests
task37b	Primary	: task.language_feature_tests
task38	Primary	: task.language_feature_tests
task39	Primary	: task.language_feature_tests
task40	Primary	: task.language_feature_tests
task41	Primary	: task.language_feature_tests
task42	Primary	: task.language_feature_tests
task43	Primary	: task.language_feature_tests
task44a	Primary	: task.language_feature_tests
task44b	Primary	: task.language_feature_tests

task45a	Primary	: task.language_feature_tests
task45b	Primary	: task.language_feature_tests
task46	Primary	: task.language_feature_tests
task46x	Primary	: task.language_feature_tests
task47	Primary	: task.language_feature_tests
task48	Primary	: task.language_feature_tests
task49	Primary	: task.language_feature_tests
task50	Primary	: task.language_feature_tests
task51	Primary	: task.language_feature_tests
task52	Primary	: task.language_feature_tests
task53	Primary	: task.language_feature_tests
task54	Primary	: withdrawn.tests
task54_mod	Primary	: task.storage_size
task55	Primary	: withdrawn.tests
task55_mod	Primary	: task.storage_size
task56	Primary	: task.storage_size
task57	Primary	: task.language_feature_tests
task58	Primary	: task.language_feature_tests
task59	Primary	: task.language_feature_tests
task60	Primary	: task.language_feature_tests
task_num_1	Primary	: task.rendezvous
task_num_5	Primary	: task.rendezvous
task_num_10	Primary	: task.rendezvous
task_num_15	Primary	: task.rendezvous
task_num_20	Primary	: task.rendezvous
task_num_25	Primary	: task.rendezvous
task_num_30	Primary	: task.rendezvous
task2_num_1	Primary	: task.rendezvous
task2_num_5	Primary	: task.rendezvous
task2_num_10	Primary	: task.rendezvous
task2_num_15	Primary	: task.rendezvous
task2_num_20	Primary	: task.rendezvous
task2_num_25	Primary	: task.rendezvous
task2_num_30	Primary	: task.rendezvous
trie1	Primary	: application.trie
	Secondary	: access.operations
trie2	Primary	: application.trie
	Secondary	: access.operations
unreach	Primary	: optimization.unreachable_code

whet1	Primary : classical.whetstone
	Incidental : math.function.arctan
	math.function.cos
	math.function.exp
	math.function.log
	math.function.sin
	math.function.sqrt
whet2	Primary : classical.whetstone
	Incidental : math.function.arctan
	math.function.cos
	math.function.exp
	math.function.log
	math.function.sin
	math.function.sqrt
whet3	Primary : classical.whetstone
	Incidental : math.function.arctan
	math.function.cos
	math.function.exp
	math.function.log
	math.function.sin
	math.function.sqrt
whet4	Primary : classical.whetstone
	Incidental : math.function.arctan
	math.function.cos
	math.function.exp
	math.function.log
	math.function.sin
	math.function.sqrt

5.7 Appendix VII, SYSTEM DEPENDENT TEST PROBLEMS

This appendix contains a list of test problems which exercise system dependent features. The test problems are listed in alphabetical order under the system dependent feature (which is also alphabetically listed). This appendix also contains a list of test program files that WITH the MATH package.

System Dependency	Test Problem Name(s)		
32 bit integers	kalman kernel16 loop4a ss258 .. ss285	kernel13 kernel16_goto loop4b ss301 .. ss315	kernel14 loop4c target
Double precision reals File I/O	gamm2 loop15 io1 .. io23	kalman ss286 .. ss303	loop3 whet3
Interface to assembly language	ss747		
Interrupts	int 0 .. int 9		
Length Clause	ew rec_glob_c* ss162 .. ss167 task54	rec_coll_c* rec_glob_u* ss242 .. ss250 task55	rec_coll_u* ss687 .. ss741 task56
MATH package	forward_euler1 kalman kernel17 .. 24 runge ss31 .. ss34 ss279 ss304 .. ss308 ss562 ss543x sim_emrpm* sim_qmpitch* sim_umnav* whet2	forward_euler2 kernel1 .. 16 loop7 .. loop8 ss14 .. ss16 ss50 .. ss51 ss291 ss406 ss586 ss650 sim_hmproto* sim_rcwfrdet* target whet4	io_80_20_1..10 kernel16_goto neural ss27 .. ss28 ss267 .. ss268 ss294 .. ss299 ss413 .. ss414 ss590 .. ss596 sim_bmbat* sim_kmdump* sim_rmkeying* whet1
Optimize = space	dhrys3	whet4	
Packing bit arrays	de-7	des7a	
Unchecked_Conversion	des7 kernel14 ss506	des7a ss259	kernel13 ss500
Unchecked_Deallocation	rec_coll_c* rec_glob_u* ss741	rec_coll_u* ss162 .. ss165	rec_glob_c* ss648
Preemptive Scheduling	delay1 .. delay14 task45a	task44a task45b	task44b

* — Names have been abbreviated in order to get them to fit in the table.

The following test programs WITH package MATH:

CFA
IO_80A
IO_80B
KALMAN
KERNEL1 .. KERNEL24
LOOP7
LOOP8
NEURAL
S0000T14
S0015T29
S0030T44
S0045T59
S0258T72
S0273T85
S0286T00
S0301T15
S0394T08
S0409T23
*S0424T38
*S0439T43
S0558T74
S0575T89
S0590T97
*S0616T30
S0631T44
S0645T51
SA8TEST
SIMULATE
WHET1
WHET2
WHET4

*These files do not contain test problems requiring a MATH package.

5.8 Appendix VIII, OPTIMIZATION TECHNIQUES

Tests for Optimizations

Algebraic simplification : 29 test problems

ss44	ss47
ss48	ss49
ss50	ss51
ss61	ss62
ss63	ss64
ss65	ss66
ss67	ss73
ss74	ss218
ss220	ss221
ss319	ss320
ss321	ss322
ss432	ss433
ss434	ss435
ss436	ss437
ss560	

Boolean variable elimination : 1 test problem

ss176

Bounds check : 6 test problems

ss174	ss192
ss193	ss194
ss255	ss368

Tests for Optimizations

Common subexpression elimination : 15 test problems

common	ss75
ss76	ss170
ss172	ss210
ss406	ss428
ss508	ss509
ss530	ss533
ss554	ss643
ss644	

Short delays : 8 test problems

delay1	delay2
delay3	delay4
delay8	delay9
delay10	delay11

Data flow : 7 test problems

ss427	ss504
ss505	ss753
ss754	ss755
ss756	

Tests for Optimizations

Dead code elimination : 9 test problems

dead	ss56
ss68	ss71
ss225	ss226
ss649	ss650
ss651	

Fold : 57 test problems

fold	ss2
ss2_mod1	ss2_mod2
ss8	ss8_mod
ss41	ss41_mod
ss42	ss42_mod
ss54	
ss55	ss60
ss83	ss169
ss173	ss185
ss189	ss190
ss216	ss216_mod
ss217	ss219
ss219_mod	ss227
ss219	ss227
ss230	ss231
ss232	ss239
ss285	ss291
ss303	ss304
ss305	ss306
ss314	ss318
ss325	ss362
ss366	ss421
ss532	ss537
ss538	ss539
ss540	ss556
ss558	ss559

Tests for Optimizations

ss561x
ss564
ss587
ss589
ss591
ss806
ss808
Task29

ss563
ss565
ss588
ss590
ss595
ss807
Task28

Habermann/Nassi : 5 test problems

Task11
Task13
Task20

Task12
Task14

Inline : 12 test problems

loop4b
ss124
ss142
ss410
whet1
Whet3

loop6
ss141
ss260
ss487
whet2
Whet4

Tests for Optimizations

Jump table : 1 test problem

ss133

Jump tracing : 6 test problems

ss182

ss184

ss619

ss183

ss250

ss620

Loop flattening : 1 test problem

ss405

Loop fusion : 1 test problem

ss180

Tests for Optimizations

Loop induction : 3 test problems

ss236
ss409

ss237

Loop invariant motion : 7 test problems

invar
ss222
ss430
ss752

ss212
ss429
ss536

Loop rotation : 3 test problems

ss385
ss387

ss386

Loop unrolling : 5 test problems

ss105
ss240
ss542

ss238
ss541

Tests for Optimizations

Machine idiom : 33 test problems

idioms	ss7
ss29	ss30
ss40	ss43
ss45	ss52
ss59	ss115
ss129	ss173
ss196	ss197
ss198	ss199
ss200	ss204
ss205	ss206
ss207	ss208
ss214	ss215
ss323	ss466
ss503	ss507
ss553	ss555
ss608	ssearch
ssearch2	

Merge tests : 4 test problems

ss175	ss178
ss179	ss440

Tests for Optimizations

Nonpositive delay : 2 test problems

Task35

Task35_delta

Order of evaluation : 13 test problems

ss413

ss414

ss415

ss416

ss417

ss418

ss545

ss546

ss547

ss548

ss549

ss550

ss551

Redundant code elimination : 7 test problems

ss93

ss195

ss261

ss376

ss377

Task27

unreach

Tests for Optimizations

Register : 34 test problems

bsort1
ciqsort
dhrys2
qsort1
runge
ss262
ss264
ss307
ss388
ss408
ss442
ss510
ss512
ss534
ss606
ss609
ss611

bsort2
dhrys1
dhrys3
qsort2
ss235
ss263
ss265
ss385x
ss407
ss412
ss443
ss511
ss531
ss557
ss607
ss610
ss612

Strength : 20 test problems

auto
gamm
loop0
loop3
lu
ss15
ss188
ss279
ss424
ss426

bmt
gamm2
loop1
loop5
puzzle
ss16
ss213
ss423
ss425
strength

Tests for Optimizations

Test swapping : 2 test problems

ss438

ss439

Unreachable code elimination : 2 test problems

ss543

ss751

5.9 Appendix IX, WITHDRAWN TEST PROBLEMS

This appendix contains a list of test problems which have been withdrawn because of problem reports. Modified versions of these tests are in the current release, with the exception of SS686. The new names have an "_mod" appended to the old name. In addition, a list of test problems which have modified versions added to the second release is provided. These problems have not been withdrawn, but there are now two (or more) versions of each.

Withdrawn Tests

DHRYS1	DHRYS2	DHRYS3	
FOLD	QUEENS		
SS95	SS96	SS97	SS98
SS686			
SS724	SS725	SS726	SS727
SS728	SS729	SS730	SS731
SS732	SS734	SS735	SS736
SS737	SS738	SS739	SS740
TASK54	TASK55		

Tests with New Versions

SS2	SS2 mod1	SS2 mod2
SS8	SS8 mod	
SS41	SS41 mod	
SS42	SS42 mod	
SS216	SS216 mod	
SS219	SS219 mod	