

AD-A182 192

SIMULATION AND ANALYSIS OF FLIGHT DECK OPERATIONS ON AN
LHA(U) CENTER FOR NAVAL ANALYSES ALEXANDRIA VA MARINE
CORPS OPERATIONS ANALYSIS GROUP S M GATES JUN 87

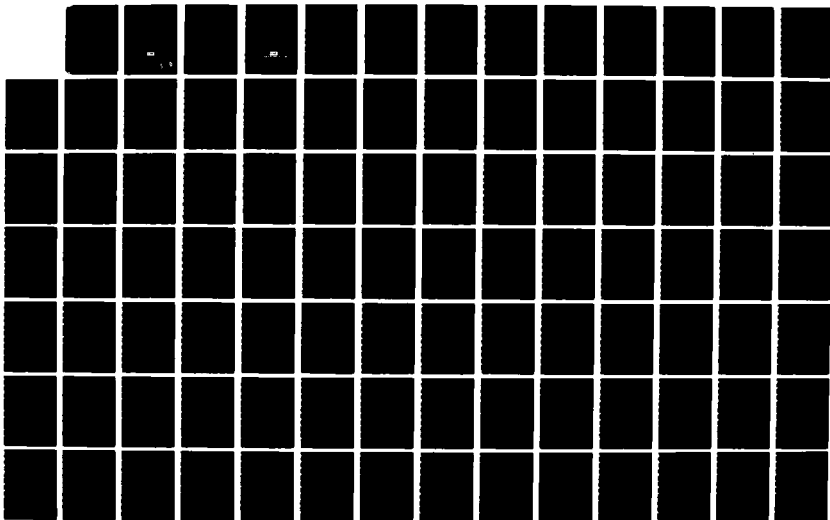
1/2

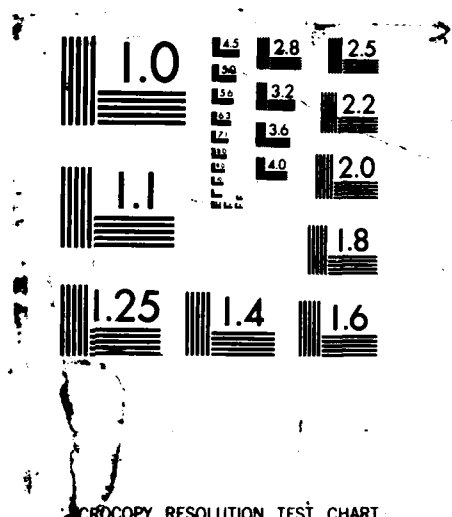
UNCLASSIFIED

CNA-PP-456 N00014-87-C-0001

F/G 1/2

NL





PHOTOCOPY RESOLUTION TEST CHART

DTIC FILE COPY

Simulation And Analysis Of Flight Deck Operations On An LHA

by

Stephen M. Gates

A Division of

CNA

Hudson Institute

CENTER FOR NAVAL ANALYSES

4401 Ford Avenue • Post Office Box 16268 • Alexandria, Virginia 22302-0268 • (703) 824-2000

This document has been approved
for public release and sales its
distribution is unlimited.,,

DTIC
ELECTE
JUL 08 1987
S D E

Cleared for Public Release. Distribution Unlimited.

The ideas expressed in this paper are those of the author.
The paper does not necessarily represent the views of the
Center for Naval Analyses, the Department of the Navy or
the Department of Defense.

Simulation And Analysis Of Flight Deck Operations On An LHA

by

Stephen M. Gates

Marine Corps Operations Analysis Group

N00014-87-C-0001



Accession For	
NTIS GRA&I	☒
DTIC TAB	☒
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

A Division of

CNA

Hudson Institute

CENTER FOR NAVAL ANALYSES

4401 Ford Avenue • Post Office Box 16268 • Alexandria, Virginia 22302-0268 • (703) 824-2000

TABLE OF CONTENTS

	<u>Page</u>
List of Illustrations	iii
List of Tables	iii
Introduction	1
Purpose	1
Background	1
Approach	2
System Description	4
Model Description	10
Event Descriptions	10
Event AC.Launched Given AC	11
Event AC.Loaded Given AC	13
Event AC.Recovered Given AC	15
Event AC.Refueled Given AC	19
Event AC.Respotted Given AC	21
Event Bone.Arrival Given AC	25
Event Deck.Arrival Given AC	28
Event Deck.Decision Given Flight	31
Event Delta.Arrival Given AC	36
Event Delta.Update.Chk	40
Event Elevator.Arrival Given AC	42
Event Flight.Check Given Flight	44
Event Flight.Launch Given Flight	46
Event Hangar.Arrival Given AC	48
Event Spot.Emergency Given AC	51
Event Spot.Open	53
Event Spot.Priority Given AC	57
Description of Entities and their Attributes	59
Measures of Effectiveness	60
Analysis of the AV-8/Helo Mix Problem	64

LIST OF ILLUSTRATIONS

		<u>Page</u>
1	LHA Flight Deck	5
2	AV-8 Flow Through System	8
3	Helicopter Flow Through System	9
4	Buildup Rate During a Six-Hour Assault (CH-46)	72
5	Buildup Rate During a Six-Hour Assault (AV-8)	73
6	Buildup Rate During a Six-Hour Assault (CH-53)	74

LIST OF TABLES

		<u>Page</u>
1	Comparison of the Planned and Actual Launch Times	61
2	Histogram Data	62
3	Measures of Effectiveness	63
4	Model Inputs for Aircraft Types	65
5	Distribution and Random Number Stream Assignments	66
6	Initial Aircraft Locations and Statuses	67
7	Flight Schedule for 6, 8, 10, and 12 AV-8 Cases	68
8	Selected Results After Ten Replicates	70
9	Paired-Difference Analysis for the 8 and 10 AV-8 Cases	71

Chapter 1

INTRODUCTION

1.1 PURPOSE

➤ The objective of this thesis was to develop a model that simulates the interactions between a number of close air support aircraft and helicopters on a flight deck during an amphibious assault. Application of this model permits a quantitative investigation into the operations of various mixes of current aircraft on a specific flight deck, and with minor modification, the effects of future aircraft and future flight decks can be examined.

The model is currently configured for operations aboard an LHA flight deck with two types of transport helicopters and the AV-8 Harrier attack aircraft. The model application presented here considers the maximum number of AV-8 Harriers that could operate effectively with a composite helicopter squadron during the amphibious assault of a Marine Amphibious Unit.

1.2 BACKGROUND

The principal mission for the United States Marine Corps is the amphibious assault. An amphibious assault is characterized by the ship-to-shore movement of troops and equipment by surface craft and aircraft to establish a beachhead and attain an operational objective. The success or failure of the assault may be determined by the initial rate of force buildup, when the landing force must rapidly be concentrated to overwhelm opposition to the landing.

Marine amphibious task forces come in a variety of sizes, ranging from a Marine Amphibious Unit (MAU) of approximately 2500 personnel to a

Marine Amphibious Force (MAF) of nearly 50,000 personnel. During peacetime, Amphibious task forces are normally deployed as MAUs embarked on 3-7 Naval amphibious ships. Usually, one of these ships will be a general purpose amphibious assault ship of the Tarawa LHA-class.

An LHA has a flight deck configured for helicopter operations, a floodable well deck for landing craft operations, a hangar deck for storing and maintaining aircraft, a 300-bed hospital, living spaces for a crew of 900 and 1900 Marines, and cargo spaces for the weapons, equipment, and supplies to support these Marines for several weeks. In addition to the composite helicopter squadron that the LHA usually operates with, a detachment of AV-8 Harriers may also be carried. AV-8s have a vertical/short takeoff and landing (V/STOL) capability, and are embarked to assist in the beach preparation prior to the assault and to provide close air support during the assault.

The AV-8s are a valuable asset to the MAU, but LHAs were not designed to support fixed wing aircraft, and helicopter operations must be temporarily suspended when AV-8s launch or recover. The competition between AV-8s and helicopters for limited flight deck assets necessarily slows the offload of troops and equipment, the delay increasing with the number of AV-8s. As mentioned earlier, speed is of the essence in the buildup ashore, but the capability afforded by the AV-8 presence can offset some reduction in the buildup rate.

In July 1982, a test was conducted in the Indian Ocean with a composite squadron of helicopters and 6 AV-8s. Among the conclusions was the finding that a composite helicopter squadron and 6 AV-8s are capable of integrated flight deck operations. Further tests have been postponed pending the introduction of a new version of the AV-8.

1.3 APPROACH

The goal of this model is to simulate the flight deck operations during an actual assault rather than an assault exercise. This approach was chosen because interest would obviously be the greatest in results obtained in a more realistic setting. For simulation purposes, the transition from an exercise to an actual assault requires the removal of the effect of training being conducted concurrently, and relaxation of the many safety requirements of peacetime operations. Exercise data must therefore be massaged with the guidance of experts from this operational area.

Two aspects of reality have not been included in the model: they are combat attrition and aircraft reliability. The intended effect of their absence is to stress the aircraft interactions throughout the simulated assault by not allowing the finite source population to decrease.

The model is a stochastic, discrete event system. The events of interest include all types of service or activity where helicopters and AV-8s might interact, both on and off the flight deck, during an assault landing.

The model verification and validation were both accomplished by stepping event by event through simulation output for many cases. A sample of this output appears in appendix B. Both efforts concluded that the model performs basically as desired.

Chapter 2

SYSTEM DESCRIPTION

The system consists of a number of aircraft (CH-46s, CH-53s, and AV-8s) that move between three major locations with a variable amount of activity and movement occurring at one of these locations.

The first major location is the Landing Zone (LZ). The LZ is the destination of the heliborne force during the assault landing. The helicopters arrive at the LZ in waves, and depart as soon as their load of troops and/or cargo has been unloaded. Their destination from the LZ is the delta pattern.

The delta pattern is a holding pattern established in the vicinity of the ship. Normally the delta pattern is a racetrack pattern around the ship, oriented on the ship's heading. Aircraft arrive to the delta pattern individually and join any other aircraft waiting there until the ship is able to recover them. The order of recovery depends on many factors, including fuel status, flying time remaining, time of next launch, and aircraft type.

The last major location is the ship; this is where the majority of the aircraft interactions occur. Relevant features of the ship include locations where aircraft may be positioned, and service activities that are not necessarily location dependent. Figure 1 displays the features of the flight deck on an LHA. The system description continues with the ship designated as of the LHA class. Locations aboard the ship are described as follows:

1. Deck Spots - There are six deck spots normally used for the launch and recovery of aircraft. These spots are located on the port side of the ship running from bow to stern. In figure 1, these spots are numbered 2,4,5,6,7,8. Spot 1 is not capable of recovering any of the aircraft listed above because of the limited amount of deck forward of the spot. Spots 3 and 9 could be

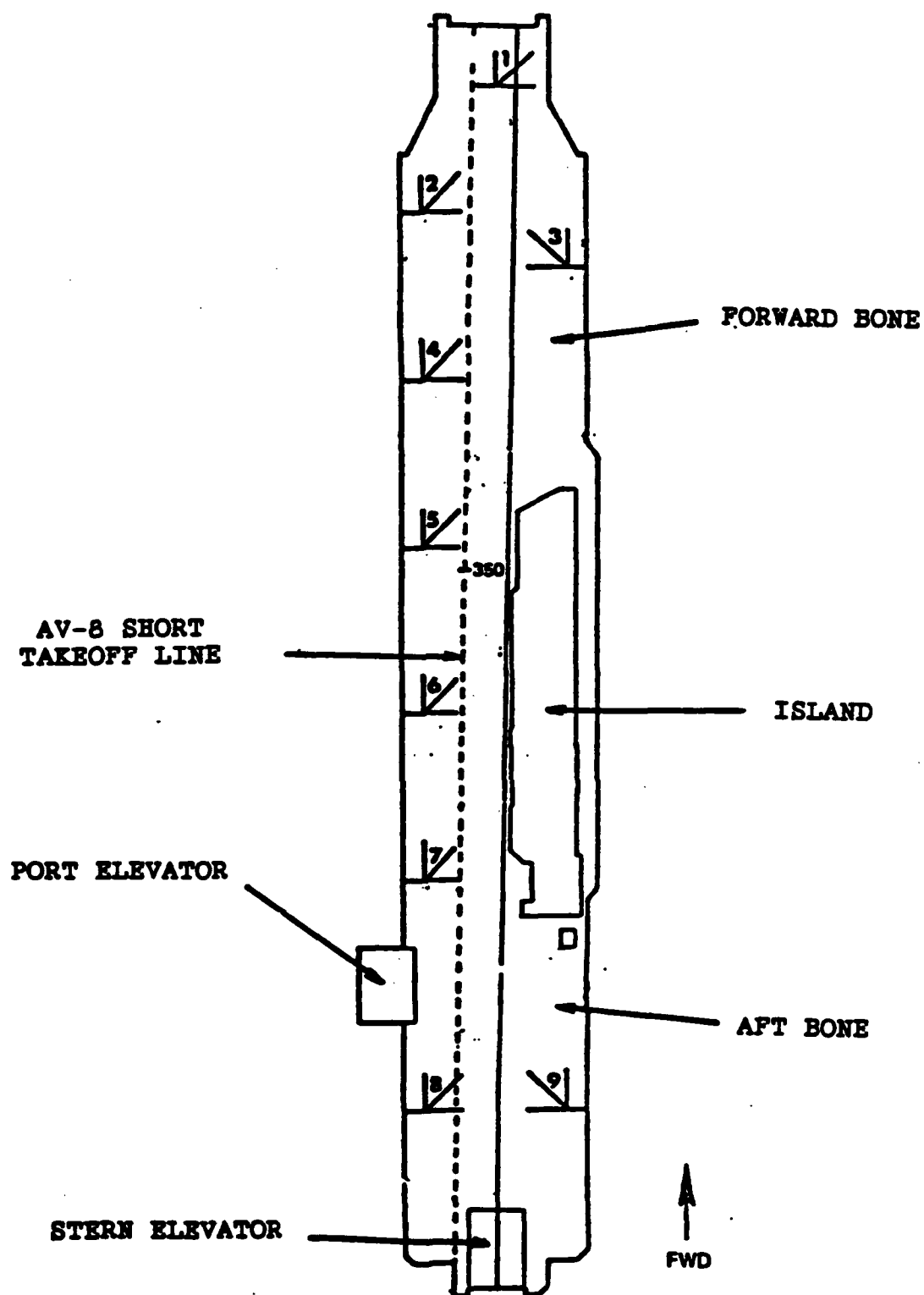


FIGURE 1. LHA FLIGHT DECK

used for launch and recovery, but they have other uses, and are called the forward and aft bones, respectively.

2. Forward Bone - The forward bone is the area forward of the island on the ship's starboard side. It is normally used as a parking area for aircraft waiting to respot to a deck spot for launching, and minor maintenance and refueling can be performed there. In the 1982 test, the forward bone was designated for use by the CH-46s.

3. Aft bone - The aft bone is the area aft of the island on the ship's starboard side. It has the same uses as the forward bone, and in the 1982 test, it was designated for use by CH-53s and AV-8s.

4. Elevator - There are two elevators on an LHA: one between spots 7 and 8 on the port side and the other centerline on the stern. The elevators connect the flight deck with the hangar deck and are capable of moving only one of the listed aircraft at a time. Normally, only one of the two elevators will be in use during an operation.

5. Hangar - The hangar is located directly below the flight deck. It is used to store aircraft and perform all levels of maintenance.

The service activities that the aircraft on the ship are affected by include:

1. Refueling - There are six refueling stations on the flight deck, each with two hoses. The stations are distributed on the deck so that an aircraft located anywhere on the deck is within the reach of at least one station's hoses. The stations are manned by four refueling crews that move between stations as necessary. The maximum number of aircraft that can refuel simultaneously is, therefore, four.

2. Loading - Helicopters are loaded on any of the six deck spots immediately prior to launching. Each spot has a combat cargo representative to oversee the loading process, so all six spots may load simultaneously. AV-8s are loaded with ordnance in the aft bone. Two ordnance teams are normally embarked, so only two AV-8s may be loaded at the same time. Ordnance can be assembled in advance and staged to the loading area, so it is assumed that ordnance is always available when requested.

3. Respotting - Respotting occurs for all aircraft to and from the deck spots, the bones, the elevator, and the hangar. Helicopters always require the assistance of a tug (tractor) when respotting. AV-8s may respot to the takeoff line from the bone, and from their recovery spot to the bone under their own power. In all other cases, AV-8s also require a tug for respotting. There are four tugs located on the flight deck and one located on the hangar

deck.

4. Launching - Helicopters launch from the six deck spots previously mentioned. AV-8s launch with with a deck run along the short takeoff line. The length of the deck run depends on the aircraft weight, the wind speed, and the temperature, but at the very least, spots 1,2, and 4 must be cleared of aircraft prior to an AV-8 launch. Aircraft launch one at a time with a small interval between launches of aircraft in a wave.

5. Recovery - All aircraft use the six deck spots previously mentioned for recovery. Generally, the smaller aircraft (CH-46s) are recovered on the forward spots and the larger aircraft (CH-53s) are recovered on the aft spots. During an emergency, all aircraft may be recovered on any of the port spots. Aircraft recover one at a time with a small interval between recoveries. Recoveries normally follow the launching of a wave.

6. Elevator service - Described above

Distributions for these service activities are presented in the discussion of Chapter 4.

The normal flow of helicopters and AV-8s through the system is shown in figures 2 and 3.

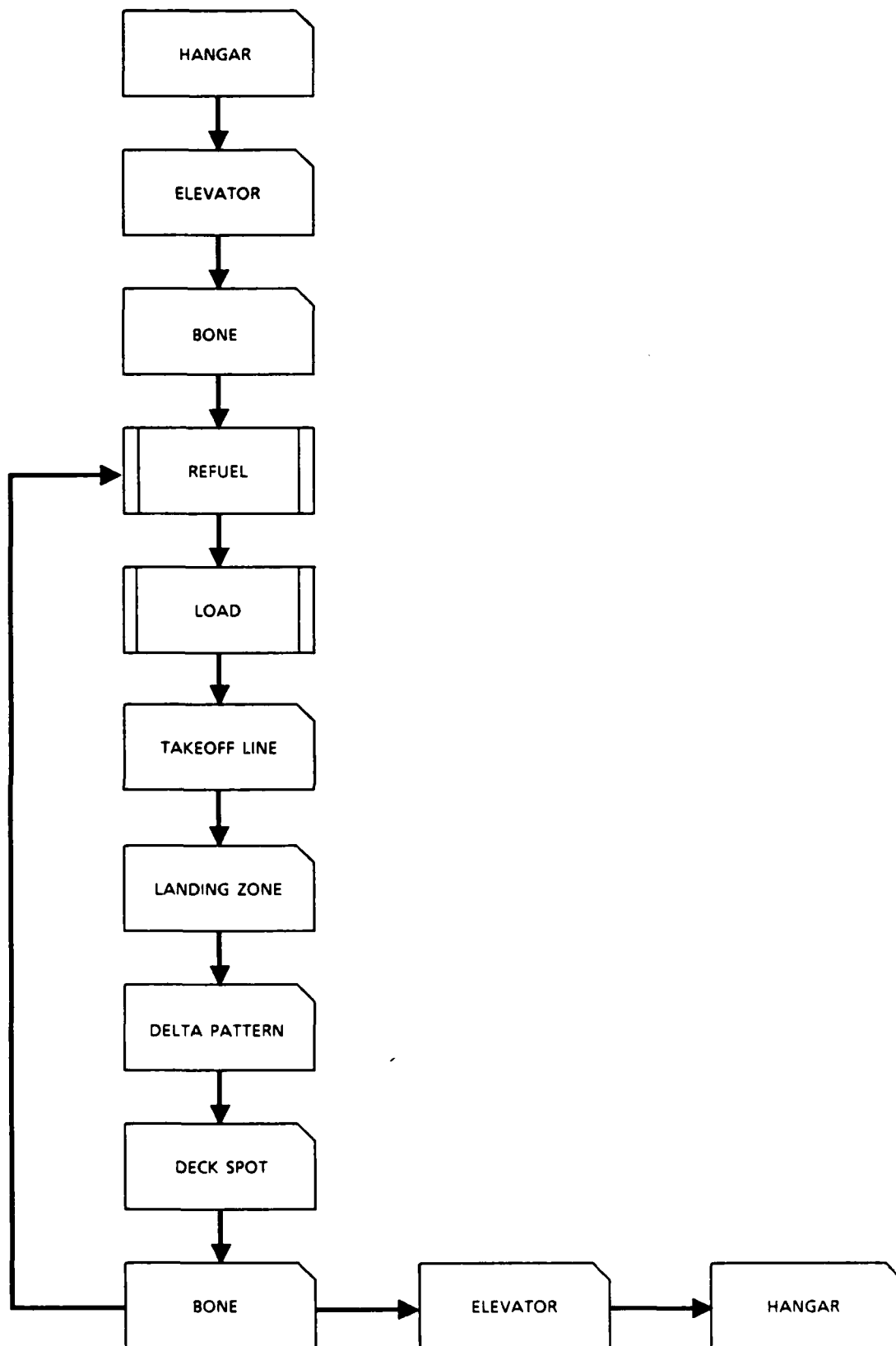


FIGURE 2. AV-8 FLOW THROUGH SYSTEM

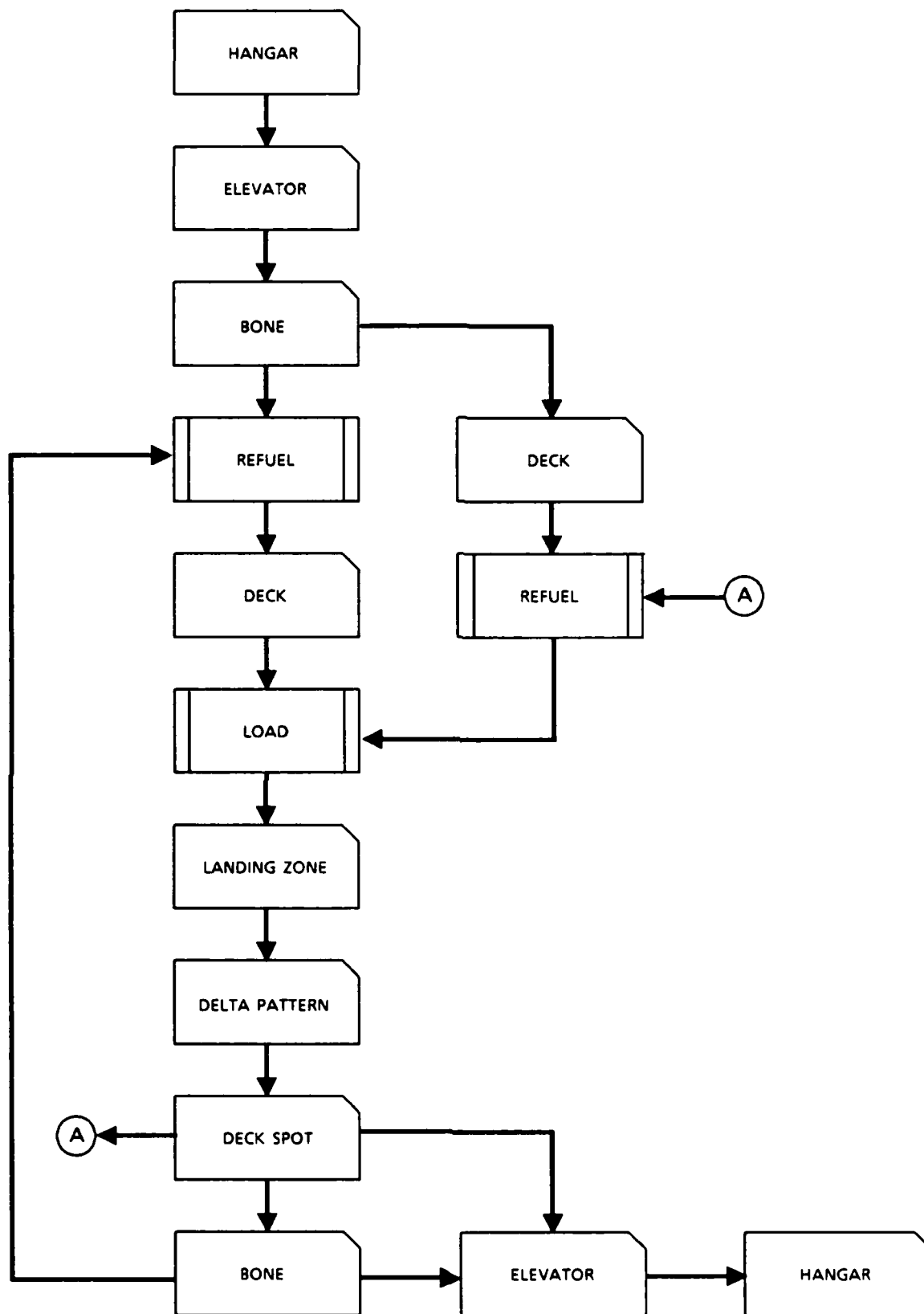


FIGURE 3. HELICOPTER FLOW THROUGH SYSTEM

Chapter 3

MODEL DESCRIPTION

The model was coded in Simscript II.5 and run on a VAX 11/785. A model listing appears in appendix A. This chapter describes how the model works with a section that details the model events, a section that defines the model entities and their attributes, and a section that presents potential measures of effectiveness that are currently in the model.

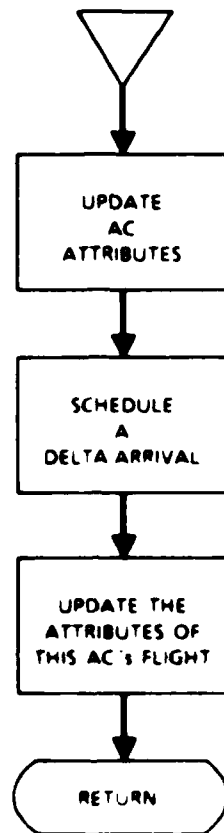
3.1 EVENT DESCRIPTIONS

In this section, the model events are described. Each subsection addresses one event with a text description followed by a flowchart. The order in which the events appear is alphabetical.

Several comments concerning conventions and symbols in the flowcharts may be helpful to the reader. There is a distinction between AC and A/C; AC refers to the aircraft that has been passed to the event, whereas A/C refers to all other aircraft that are treated during the event process. The triangular node indicates the beginning of an event. The rectangular node represents a positive action that occurs at that point in the model, whereas the rectangular node with extra vertical lines represents a comment concerning a state or a set of actions. There are two symbols that transfer the flow to different areas of the event, signified by a node with a single letter. The first symbol is a circle, and in this case, the flow is transferred to another area on the same page. The other symbol has five sides, and transfers the flow to another page of this event. The return node indicates a state in the simulation where control is returned to the Simscript scheduler, and marks the end of activity in the current event.

3.1.1 EVENT AC.LAUNCHED GIVEN AC

The occurrence of this event signals the departure of AC from the ship. The event DELTA.ARRIVAL is scheduled by computing the time required to make the round trip to the landing zone and adding the time (random) necessary to accomplish its mission (time to unload). A deck spot becomes available, and a SPOT.OPEN event is scheduled.



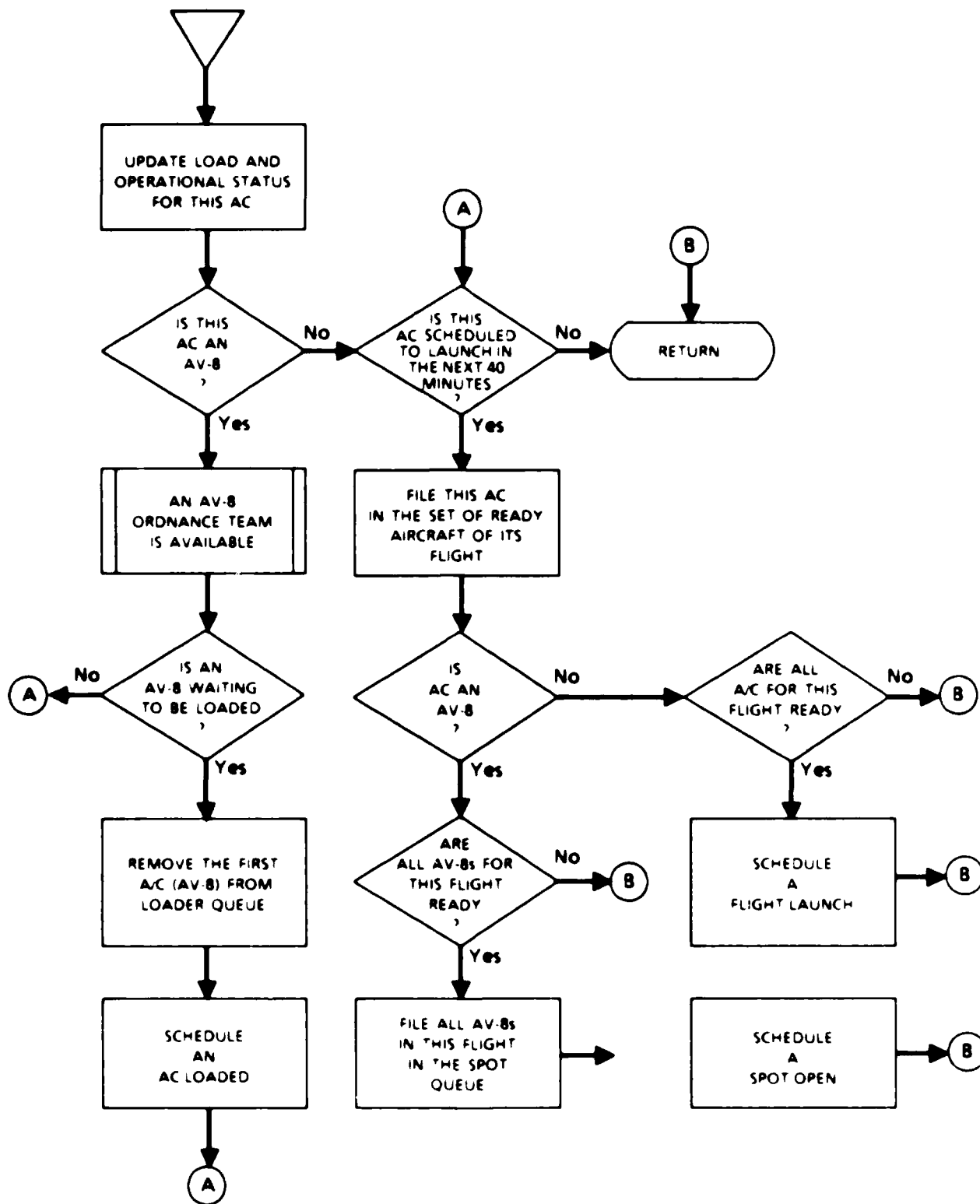
EVENT AC.LAUNCHED GIVEN AC

3.1.2 EVENT AC.LOADED GIVEN AC

The occurrence of this event signals the completion of the loading process for the aircraft AC. The action that occurs next depends on the type of aircraft that AC is and its launch time.

If AC is an AV-8, an ordnance team becomes available, and if another AV-8 is waiting for an ordnance team, - AC.LOADED is scheduled. If AC has a launch time in the next 40 minutes, and both AV-8s in this flight have now been loaded, they are ready to be respotted to the takeoff line prior to launch. This is accomplished by filing them in the spot queue and scheduling a SPOT.OPEN event.

If AC is a helicopter, has a launch time in the next 40 minutes, and all aircraft of this flight have been loaded (launch time should be less than 10 minutes away if all aircraft are loaded), then a FLIGHT.LAUNCH event is scheduled.



EVENT AC.LOADED GIVEN AC

3.1.3 EVENT AC.RECOVERED GIVEN AC

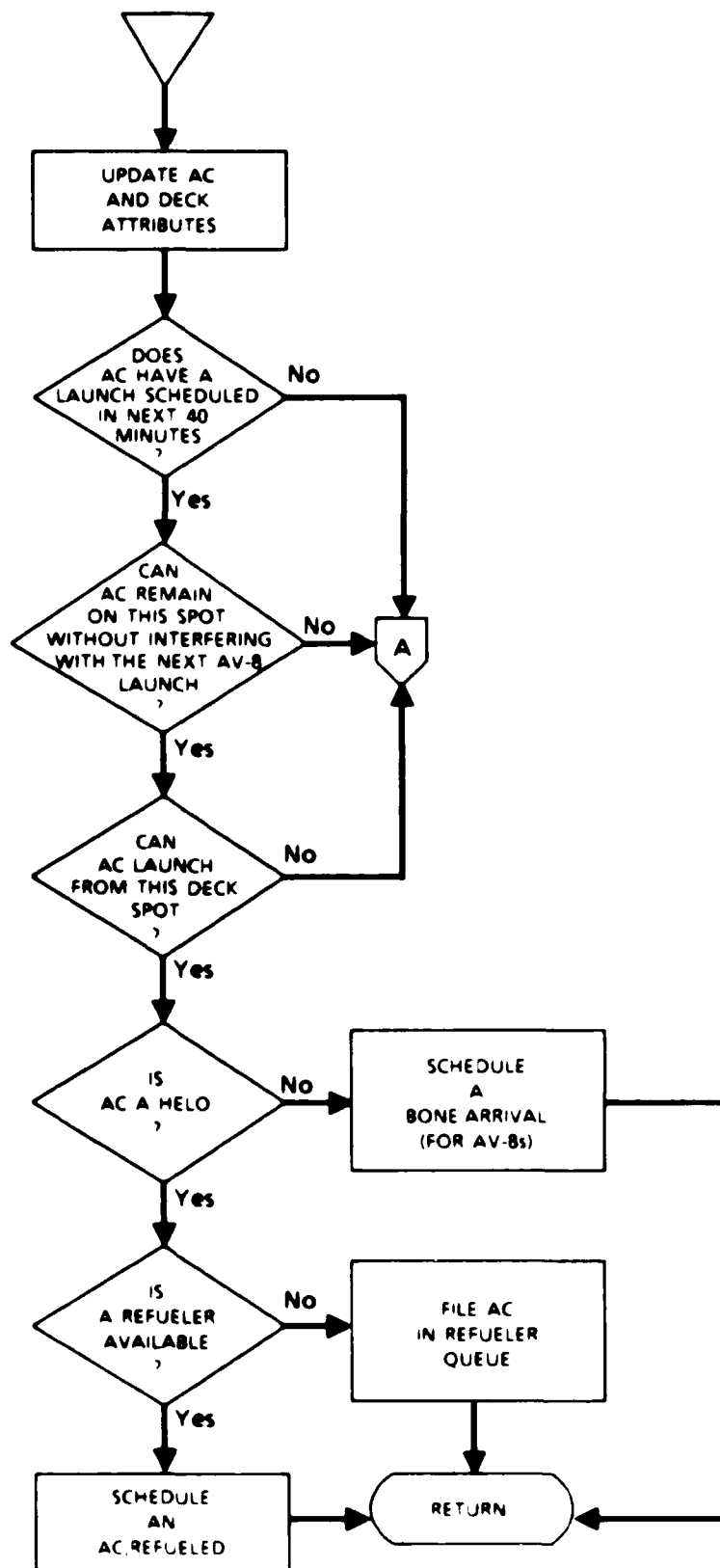
This event occurs when the aircraft AC has returned to the ship and landed on one of the six deck spots. AC will remain on this deck spot only if all of the following conditions are satisfied:

- : AC must have a launch time scheduled to occur in the next 40 minutes
- : AC's presence on this deck spot must not interfere with the next AV-8 launch
- : AC must be compatible to launch from this deck spot
- : AC must be a helicopter

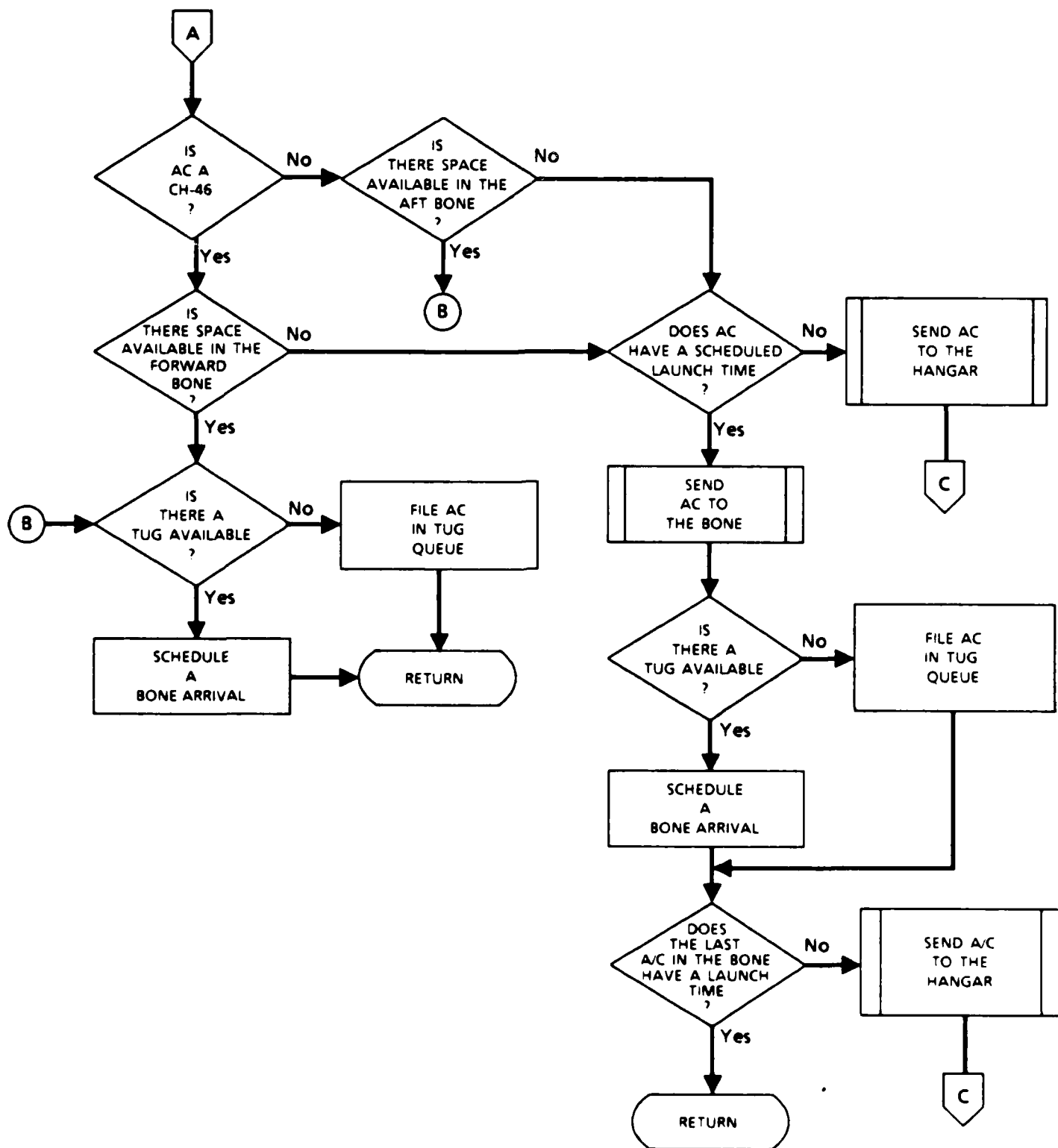
If all of these conditions are satisfied, the next step for this helicopter is to schedule an AC.REFUELED event.

If any of these conditions are not met, AC is sent to the appropriate bone if there is space in that bone. If the bone is full, AC is still sent to the bone if AC has a scheduled launch time; otherwise, AC is sent to the hangar. Helicopters require a tug in all cases for deck movement. AV-8s can respot to the bone under their own power, but require a tug when going to the hangar (elevator).

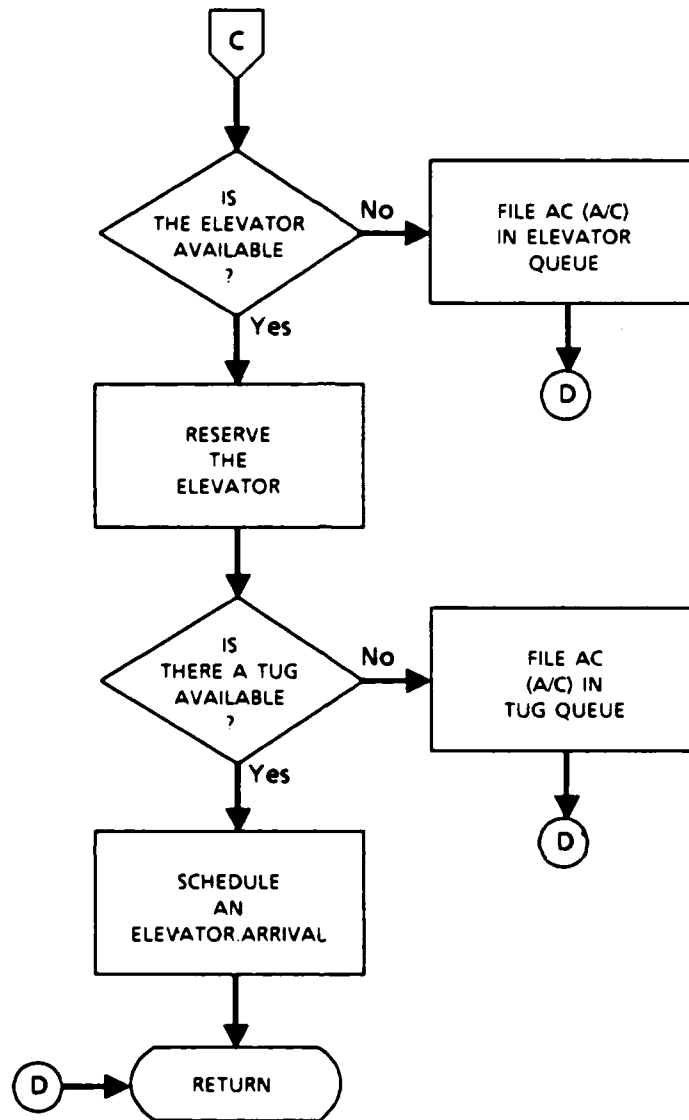
If the bone was full and AC has scheduled a BONE.ARRIVAL, the aircraft in that bone are checked for any that do not have a future launch time. If one is found, this aircraft is sent to the hangar (elevator).



EVENT AC.RECOVERED GIVEN AC



CONTINUED: EVENT AC.RECOVERED



CONTINUED: EVENT AC.RECOVERED

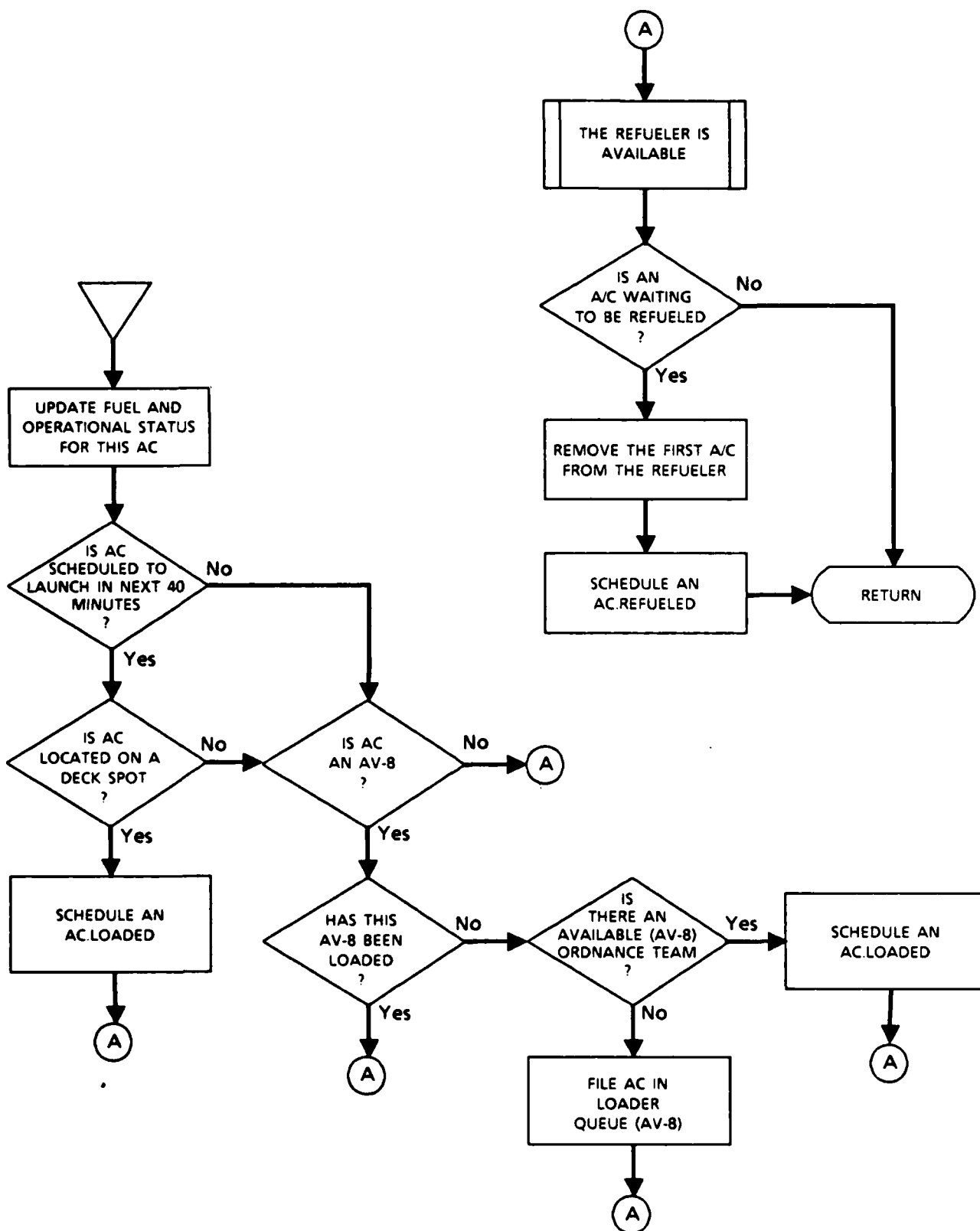
3.1.4 EVENT AC.REFUELED GIVEN AC

The occurrence of this event signals the completion of the refueling process for the aircraft AC. The action that occurs next depends on the type of aircraft that AC is and the launch time of AC.

If AC is an AV-8 and the loading of ordnance has not started, regardless of the next launch time for AC, the next step is to schedule an AC.LOADED event. Although not currently played in the model, these aircraft without launch times can be thought of as aircraft that are being readied for on-call missions.

If AC has a launch time in the next 40 minutes and is located on a deck spot, then it must be a helicopter, because AV-8s are not refueled on deck spots. The next step for this helicopter is to schedule an AC.LOADED event. The loading event will be scheduled to occur no more than 10 minutes prior to the scheduled launch time.

In all cases, a refueler becomes available when an AC.REFUELED event occurs. If an aircraft is waiting to be refueled, it is removed from the refueler queue and an event AC.REFUELED is scheduled.



EVENT AC.REFUELED GIVEN AC

3.1.5 EVENT AC.RESPOTTED GIVEN AC

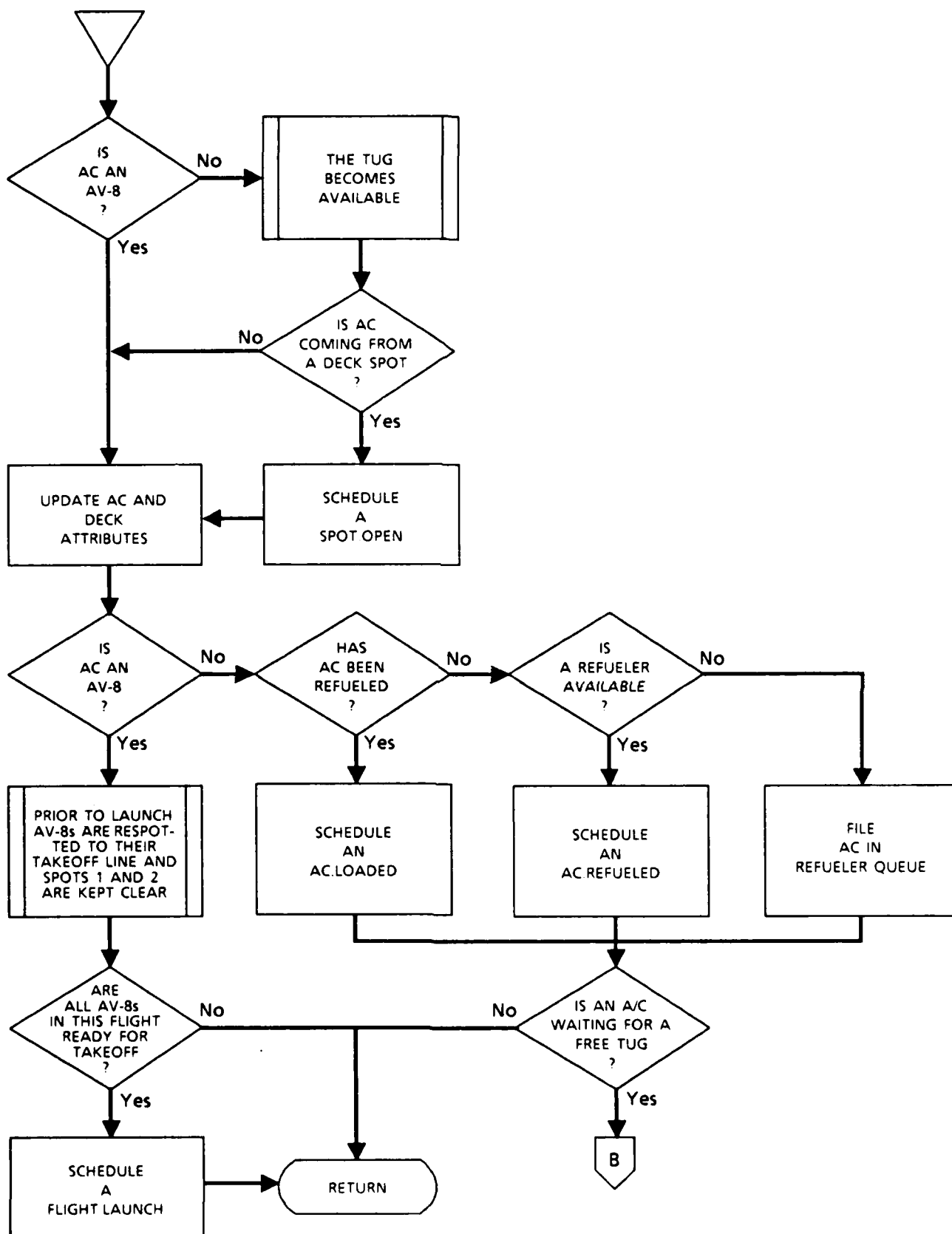
The occurrence of this event signals that AC has arrived at one of the six deck spots from which an aircraft may launch.

If AC is an AV-8, the movement has been made under its own power to the short takeoff line without a tug. The respotted AV-8 has been fueled and loaded, and is ready for takeoff. If both AV-8s in this flight are ready to launch, a FLIGHT.LAUNCH event is scheduled.

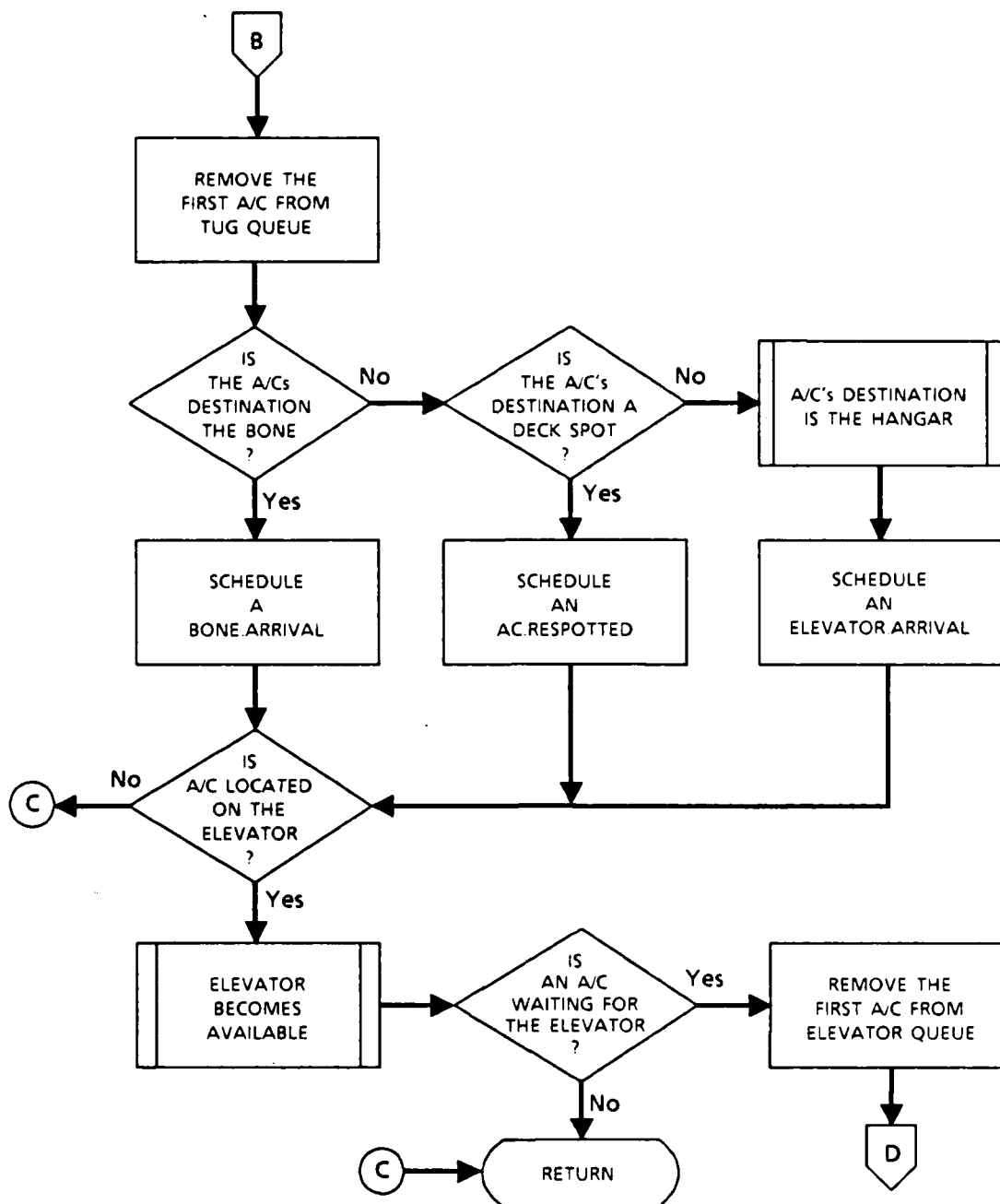
If AC is a helicopter, the movement has been made with a tug, and this tug becomes available to move another aircraft. If the helicopter has not been refueled, then the next event for this helicopter is an AC.REFUELED. If the helicopter has been refueled, the the next event scheduled will be an AC.LOADED. The loading event will be scheduled to occur no more that 10 minutes prior to the helicopter's scheduled launch time.

If there are aircraft waiting for a tug, and a tug is now free, the first aircraft in tug queue is removed from the queue and the appropriate event is scheduled to move it to its destination. If this aircraft was located on the elevator, the elevator becomes available.

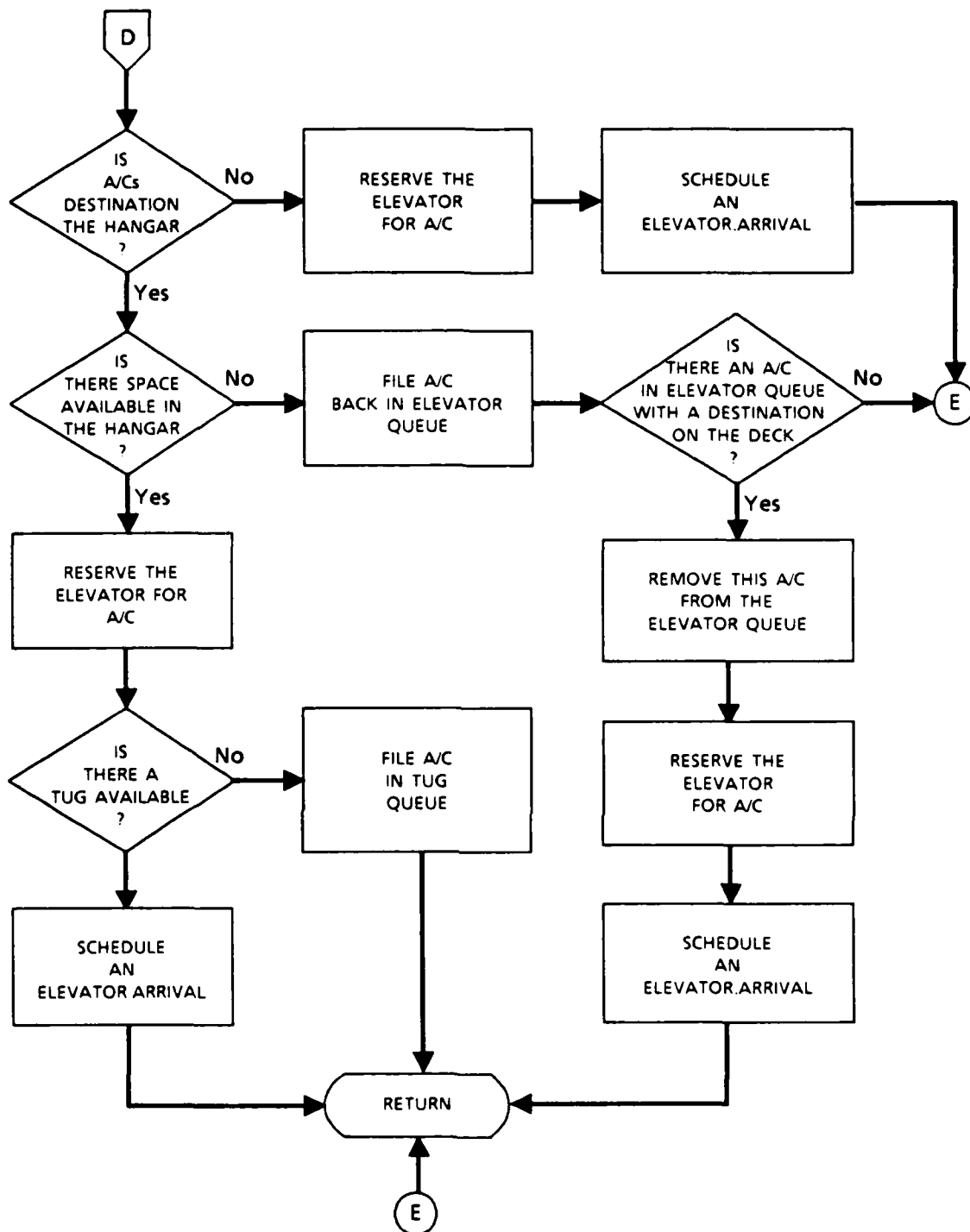
If the elevator is available and there are aircraft waiting in the elevator queue, the first aircraft in the queue whose destination is able to receive it is removed from queue, and an ELEVATOR.ARRIVAL event is scheduled.



EVENT AC.RESPOTTED GIVEN AC



CONTINUED: EVENT AC.RESPOTTED



CONTINUED: EVENT AC.RESPOTTED

3.1.6 EVENT BONE.ARRIVAL GIVEN AC

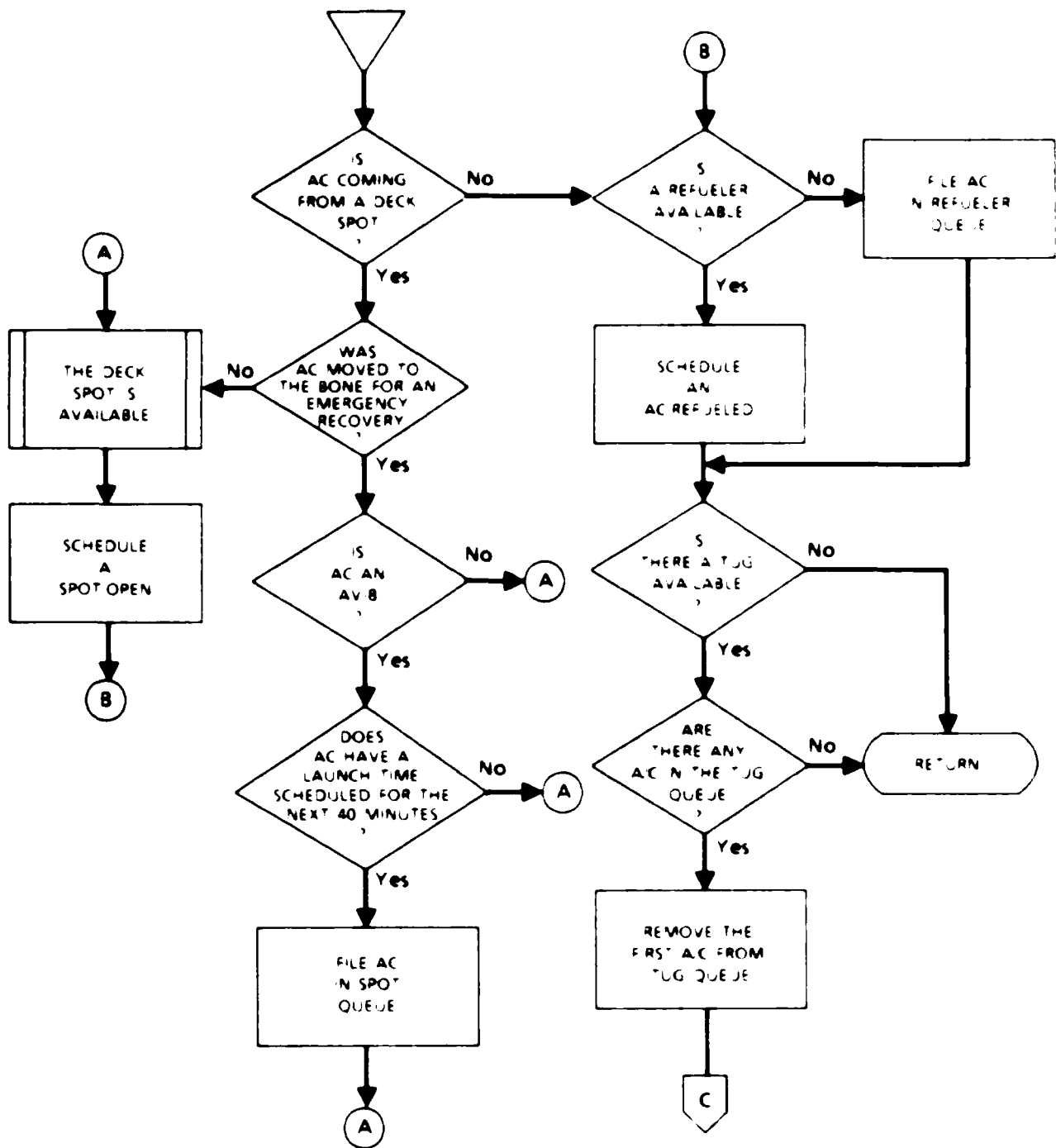
The occurrence of this event signals that AC has respotting to either the forward or aft bone.

If ACs last location was a deck spot, the spot is now available and a SPOT.OPEN event is scheduled.

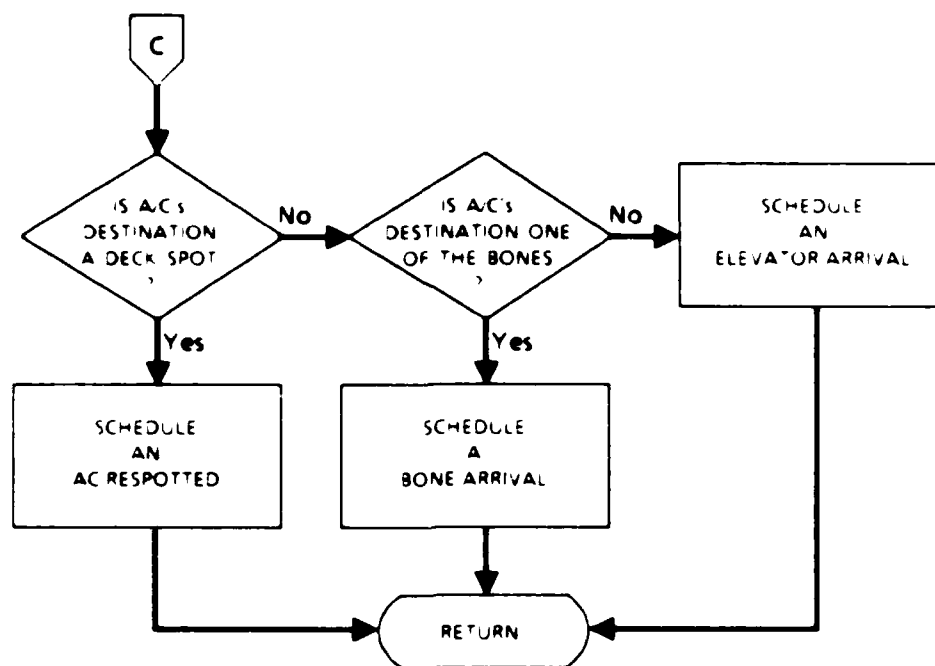
If AC requires fuel and a refueler is available, an event AC.REFUELED is scheduled. Otherwise, AC is filed in the queue to await the next available refueler.

If there are aircraft waiting for a tug, and a tug is now free, the first aircraft in tug queue is removed from the queue and the appropriate event is scheduled to move it to its destination. If this aircraft was located on the elevator, the elevator becomes available.

If the elevator is available and there are aircraft waiting in the elevator queue, the first aircraft in the queue whose destination is able to receive it is removed from queue, and an ELEVATOR.ARRIVAL event is scheduled.



EVENT BONE.ARRIVAL GIVEN AC



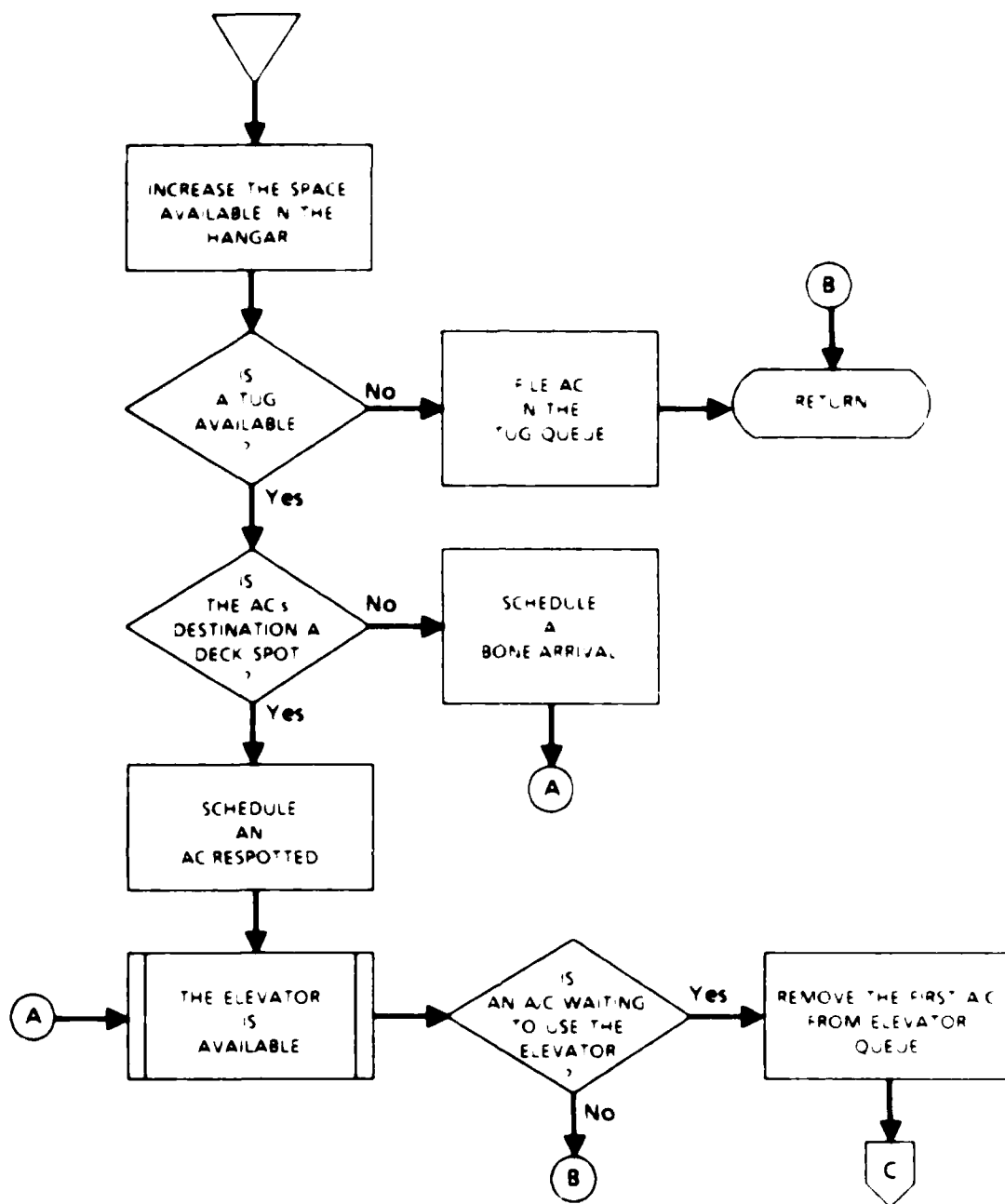
CONTINUED: EVENT BONE.ARRIVAL

3.1.7 EVENT DECK.ARRIVAL GIVEN AC

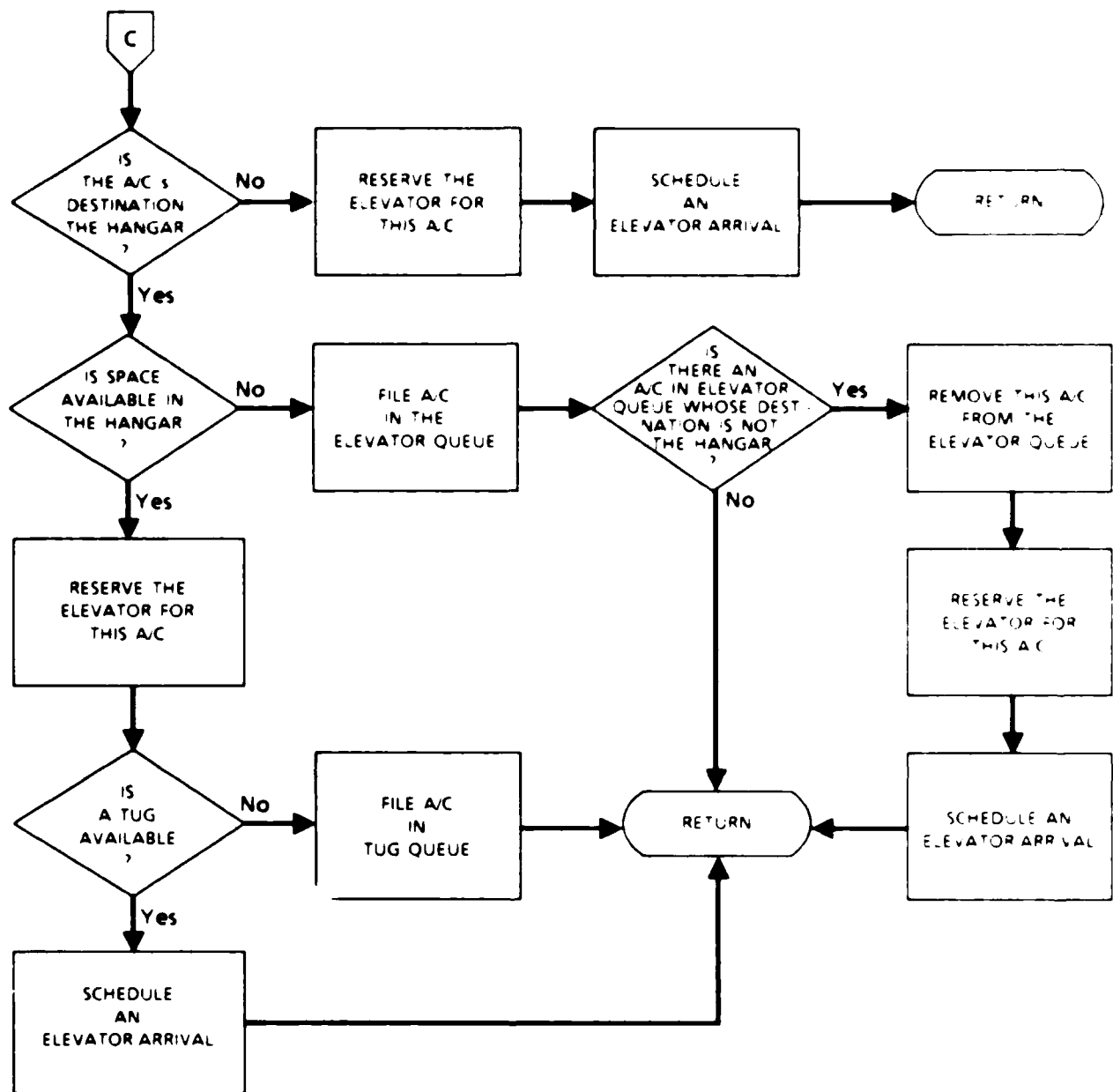
This event occurs when AC has arrived by elevator to the flight deck. In all cases, a tug is required to move AC to and from the elevator. If a tug is available, the event BONE.ARRIVAL or AC.RESPOTTED is scheduled depending on the destination of AC. Also, if a tug is available, the elevator will soon be available, and the elevator queue is checked for waiting aircraft.

If an aircraft is waiting for the elevator, and its destination is the hangar, there must be space available in the hangar. If these conditions are met, the elevator is reserved. If a tug is available, an ELEVATOR.ARRIVAL event is scheduled.

If an aircraft is waiting for the elevator, and its destination is the flight deck, the elevator is reserved and an ELEVATOR.ARRIVAL event is scheduled. (The tug on the hangar deck is assumed to be available whenever the elevator is.)



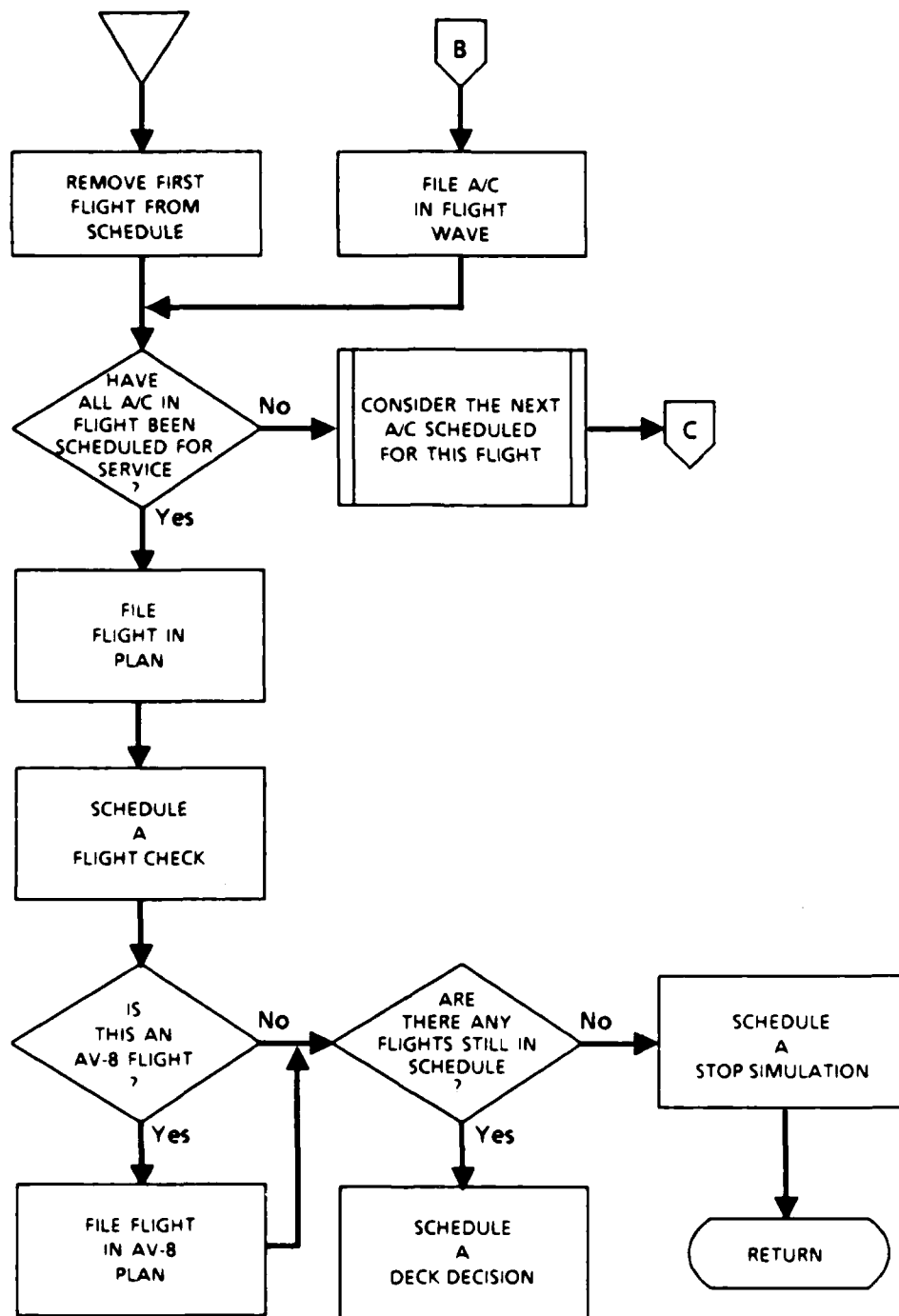
EVENT DECK.ARRIVAL GIVEN AC



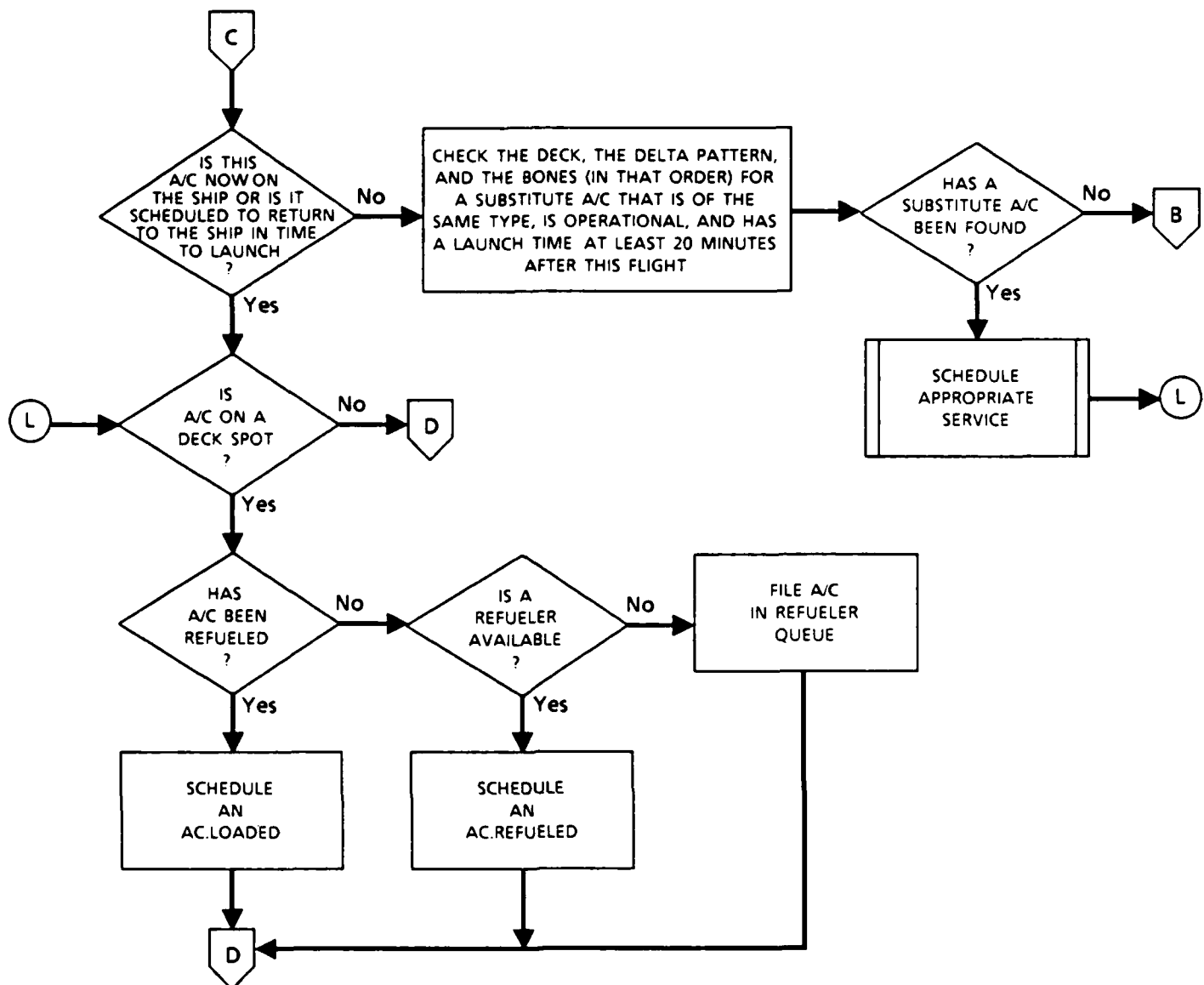
CONTINUED: EVENT DECK ARRIVAL

3.1.8 EVENT DECK.DECISION GIVEN FLIGHT

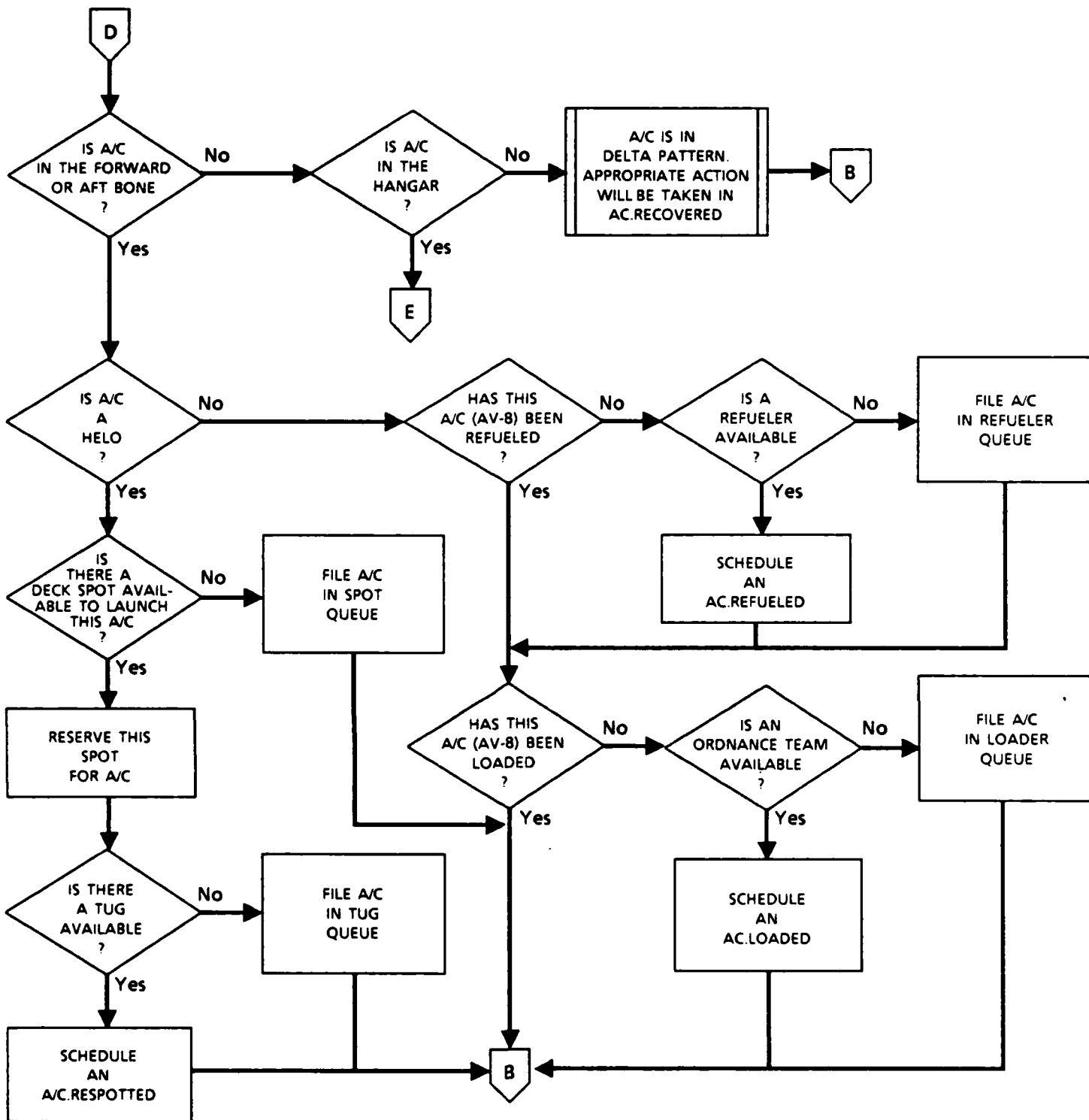
The primary function of this event is to ensure that aircraft with approaching launch times are preparing to launch. If an aircraft is not likely to be ready, it is replaced with another aircraft if one is available. The checking of aircraft statuses takes place 40 minutes prior to the scheduled launch time. If an aircraft has not begun making the necessary preparations, the appropriate event(s) are scheduled.



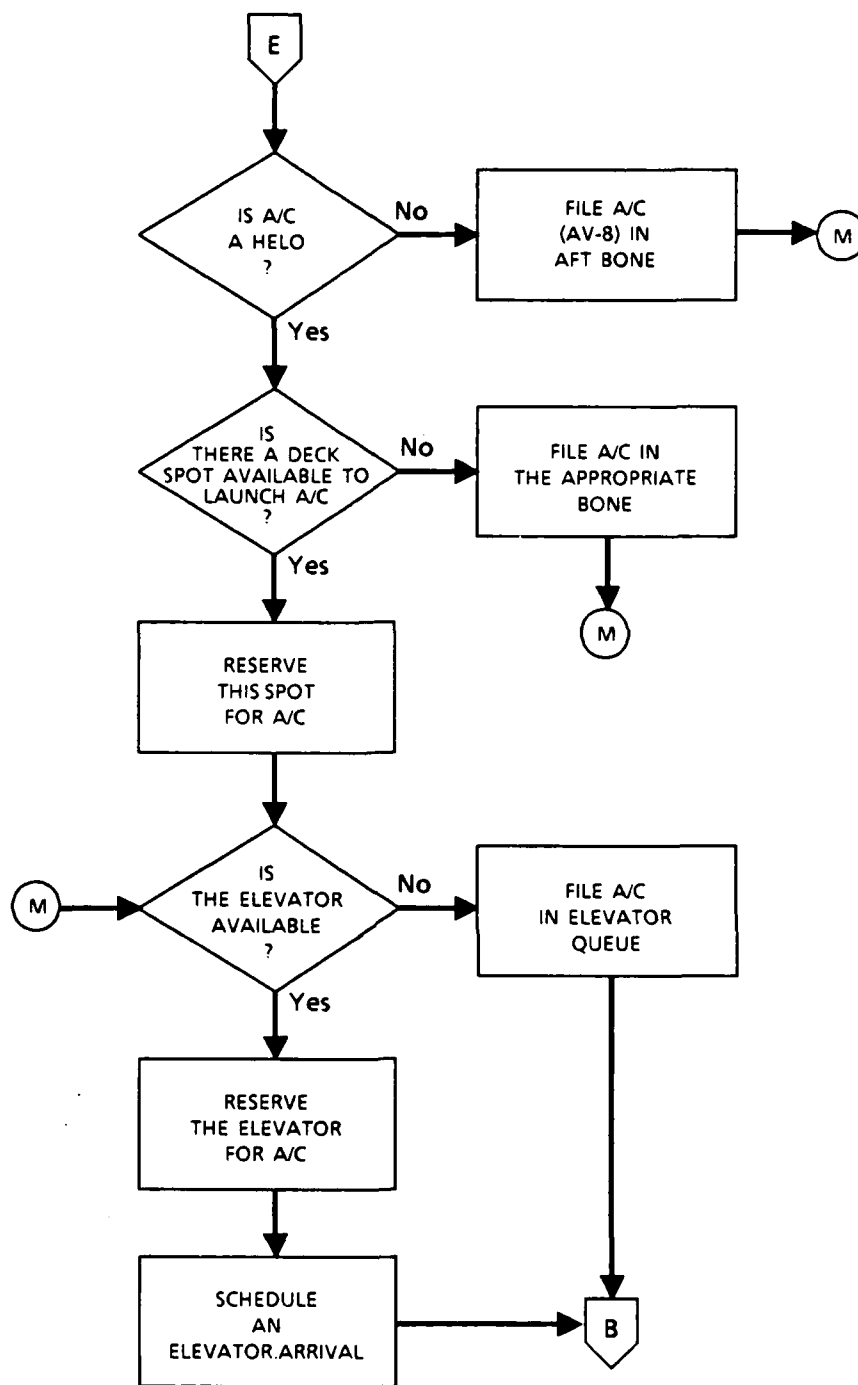
EVENT DECK.DECISION GIVEN FLIGHT



CONTINUED: EVENT DECK.DECISION



CONTINUED: EVENT DECK.DECISION



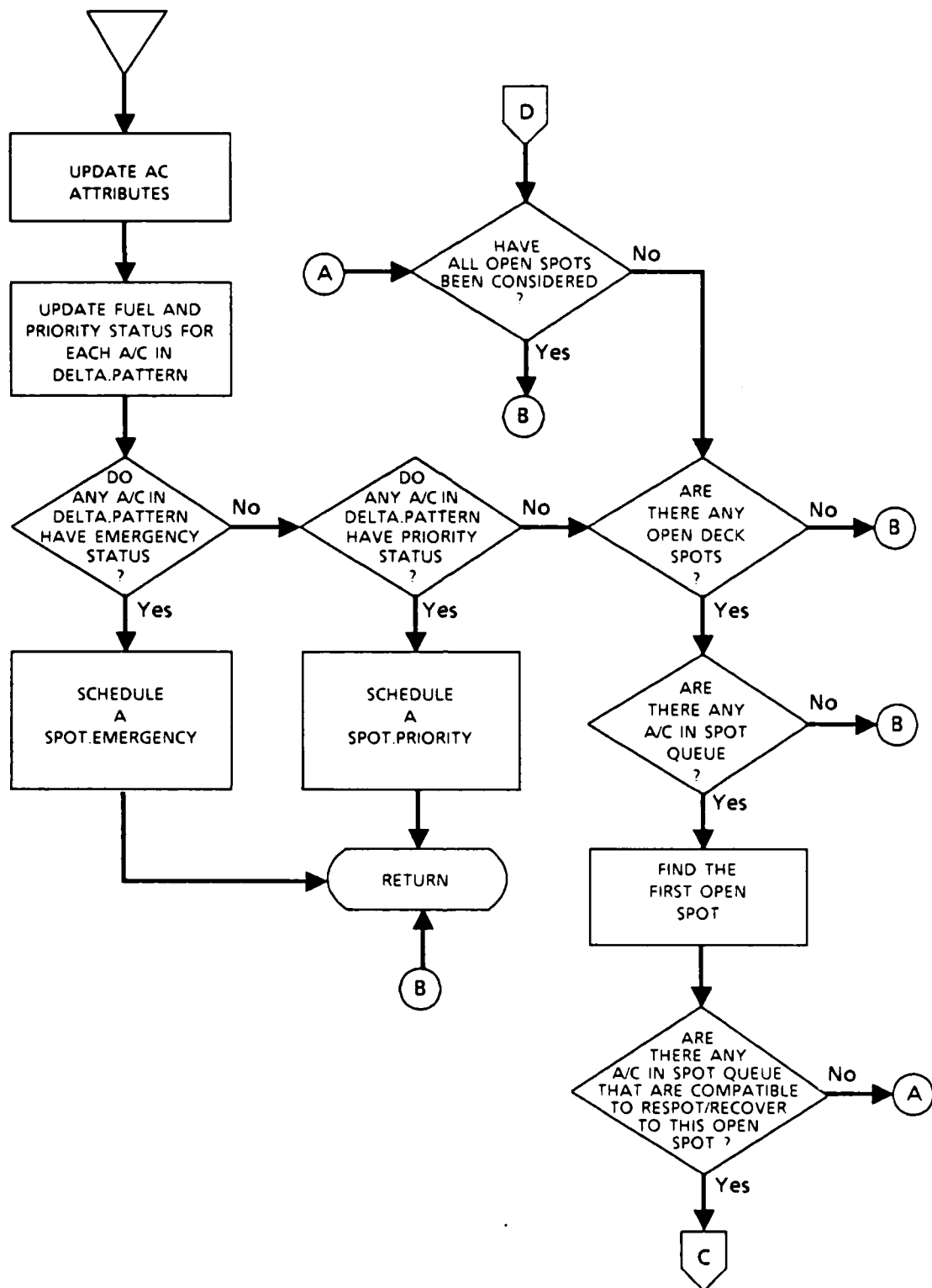
CONTINUED: EVENT DECK.DECISION

3.1.9 EVENT DELTA.ARRIVAL GIVEN AC

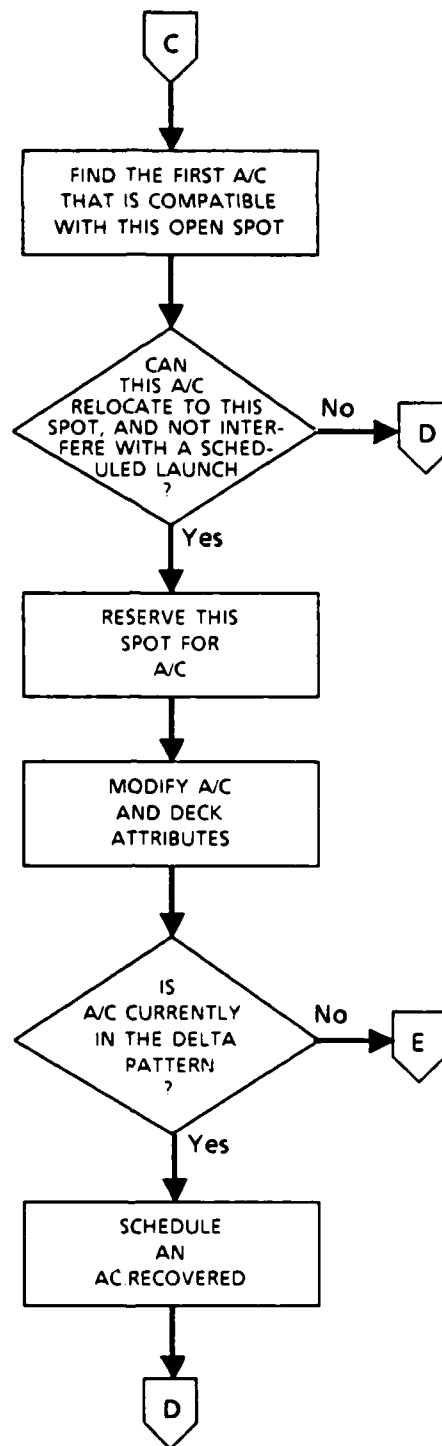
This event occurs when AC has completed its mission and returned to the ship's vicinity for recovery. AC enters the holding (delta) pattern to await clearance to land. The fuel status of each aircraft in the delta pattern is updated and checked to see if it entitles the aircraft for priority or emergency treatment.

If an aircraft is qualified for special treatment, either event SPOT.EMERGENCY or SPOT.PRIORITY , as appropriate, is scheduled.

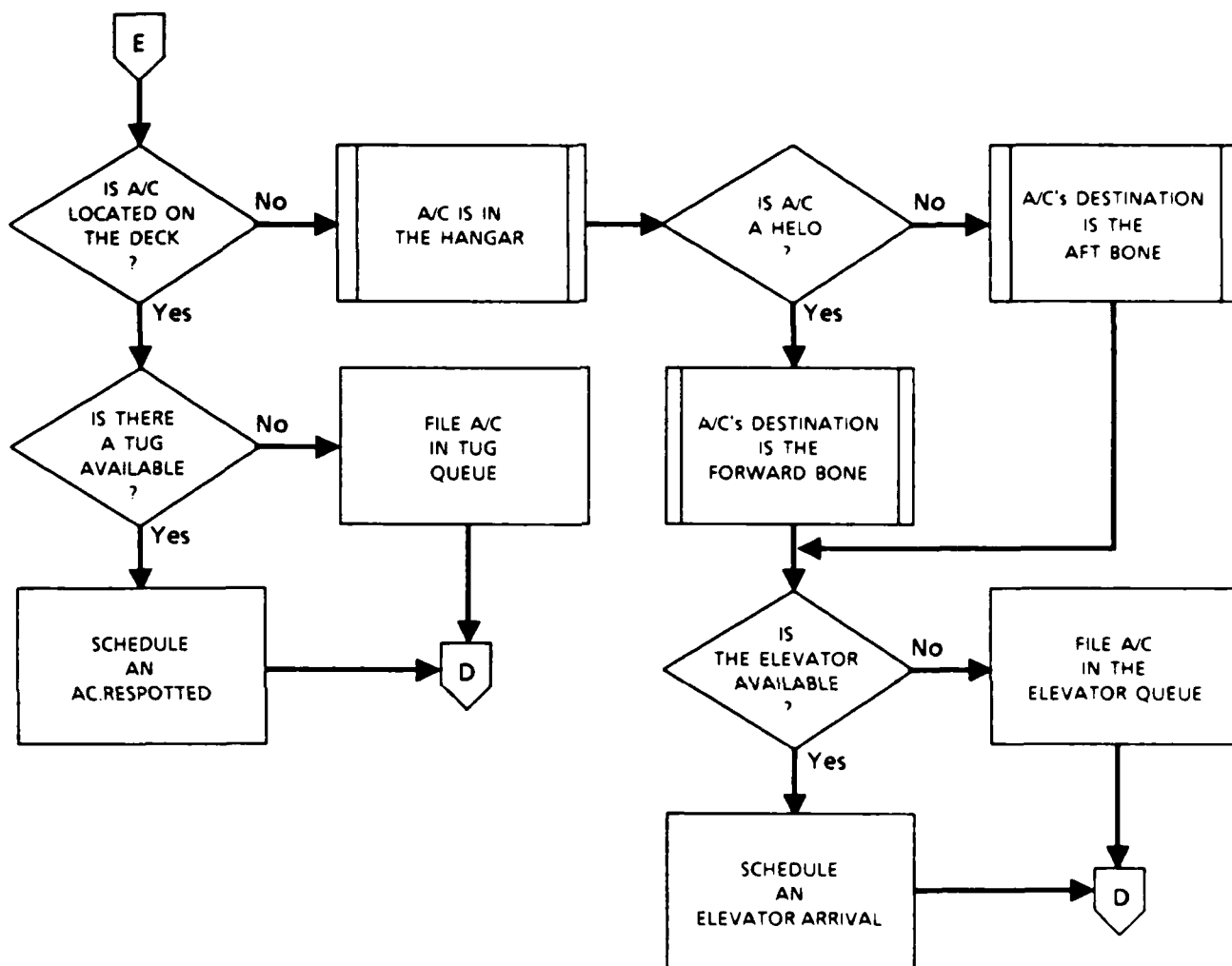
If no aircraft qualifies for special treatment, aircraft in the spot queue are considered for recovery or respotting to any open spots. Any aircraft that are compatible with an open spot, and will not interfere with another aircraft's launch are scheduled for an AC.RECOVERED or AC.RESPOTTED event.



EVENT DELTA.ARRIVAL GIVEN AC



(CONTINUED)
EVENT DELTA.ARRIVAL GIVEN AC

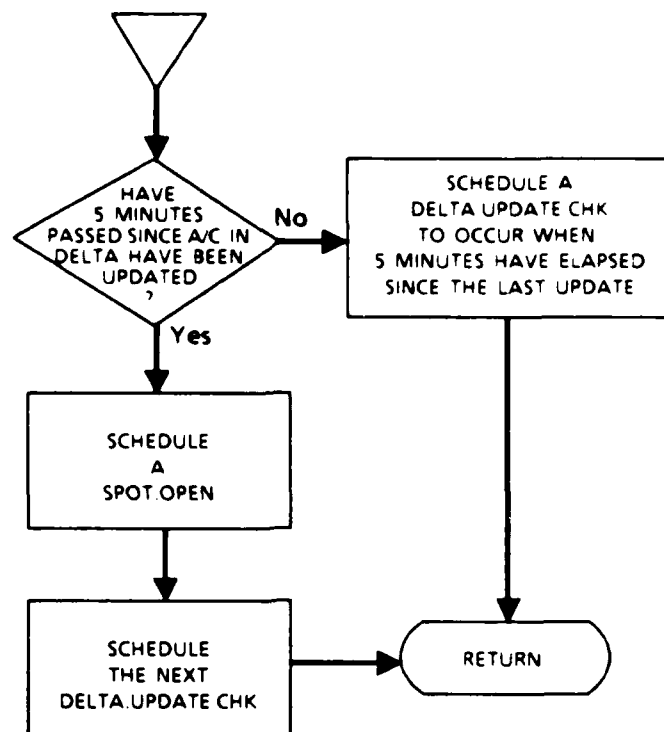


(CONTINUED) EVENT DELTA.ARRIVAL

3.1.10 EVENT DELTA.UPDATE.CHK

This event occurs at least once every five minutes. Its purpose is to ensure that the fuel levels of aircraft in the delta pattern are remaining at safe levels. It accomplishes this by ensuring that the delta update algorithm in the DELTA.ARRIVAL and SPOT.OPEN events is activated at least once every five minutes.

If five minutes have passed since the last update, a SPOT.OPEN event is scheduled.



EVENT DELTA.UPDATE.CHK

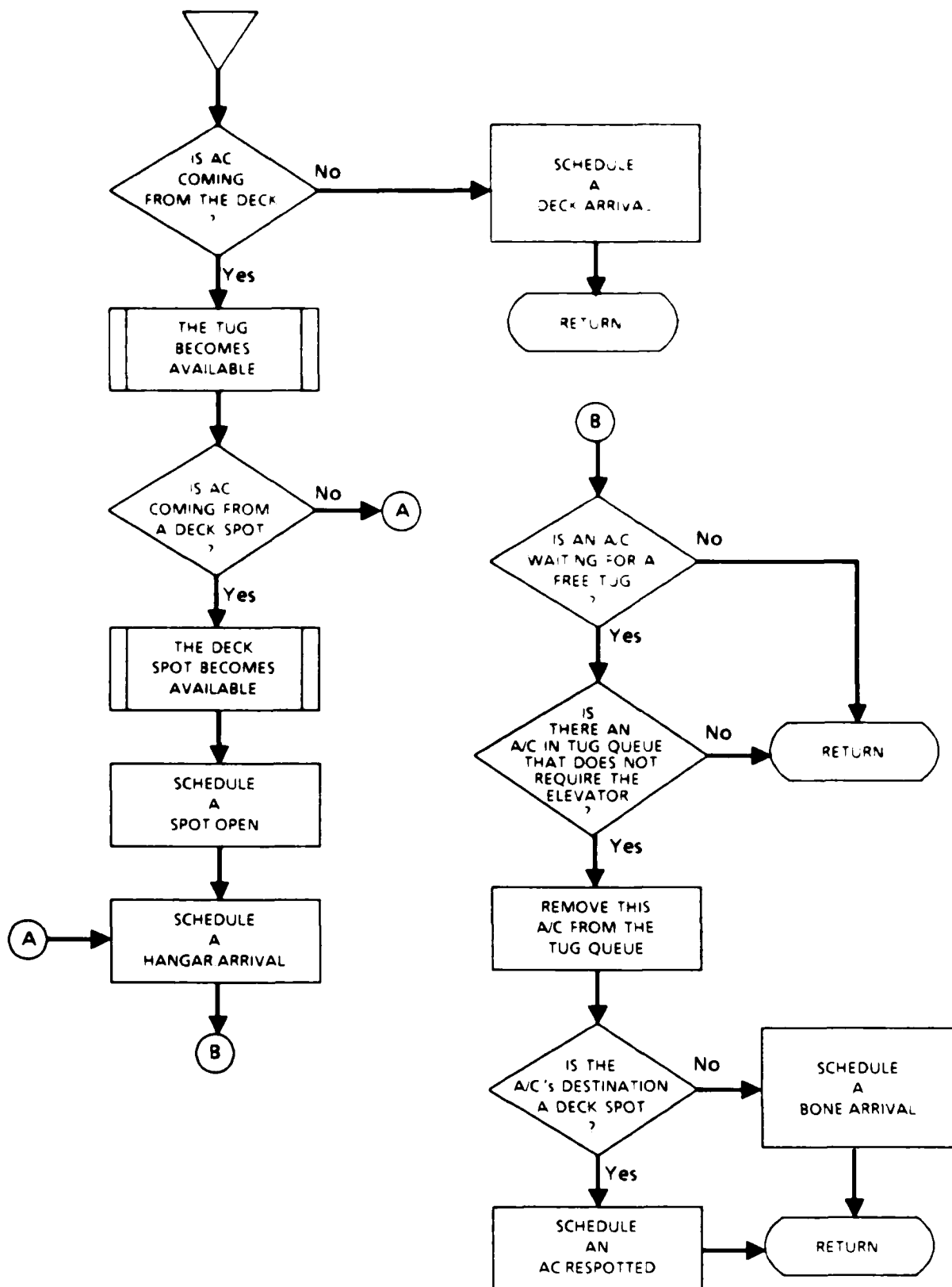
3.1.11 EVENT ELEVATOR.ARRIVAL GIVEN AC

This event occurs when the aircraft AC has been respotted to the elevator and tied down before the movement between the hangar and flight decks.

If AC is going to the hangar deck, a tug becomes available, and a HANGAR.ARRIVAL event is scheduled. If AC is coming from a deck spot, the deck spot becomes available and a SPOT.OPEN event is scheduled.

If AC is going to the flight deck, a DECK.ARRIVAL event is scheduled.

If there are aircraft waiting for a tug, and a tug is now free, the first aircraft in tug queue is removed from the queue and the appropriate event is scheduled to move it to its destination.



EVENT ELEVATOR.ARRIVAL GIVEN AC

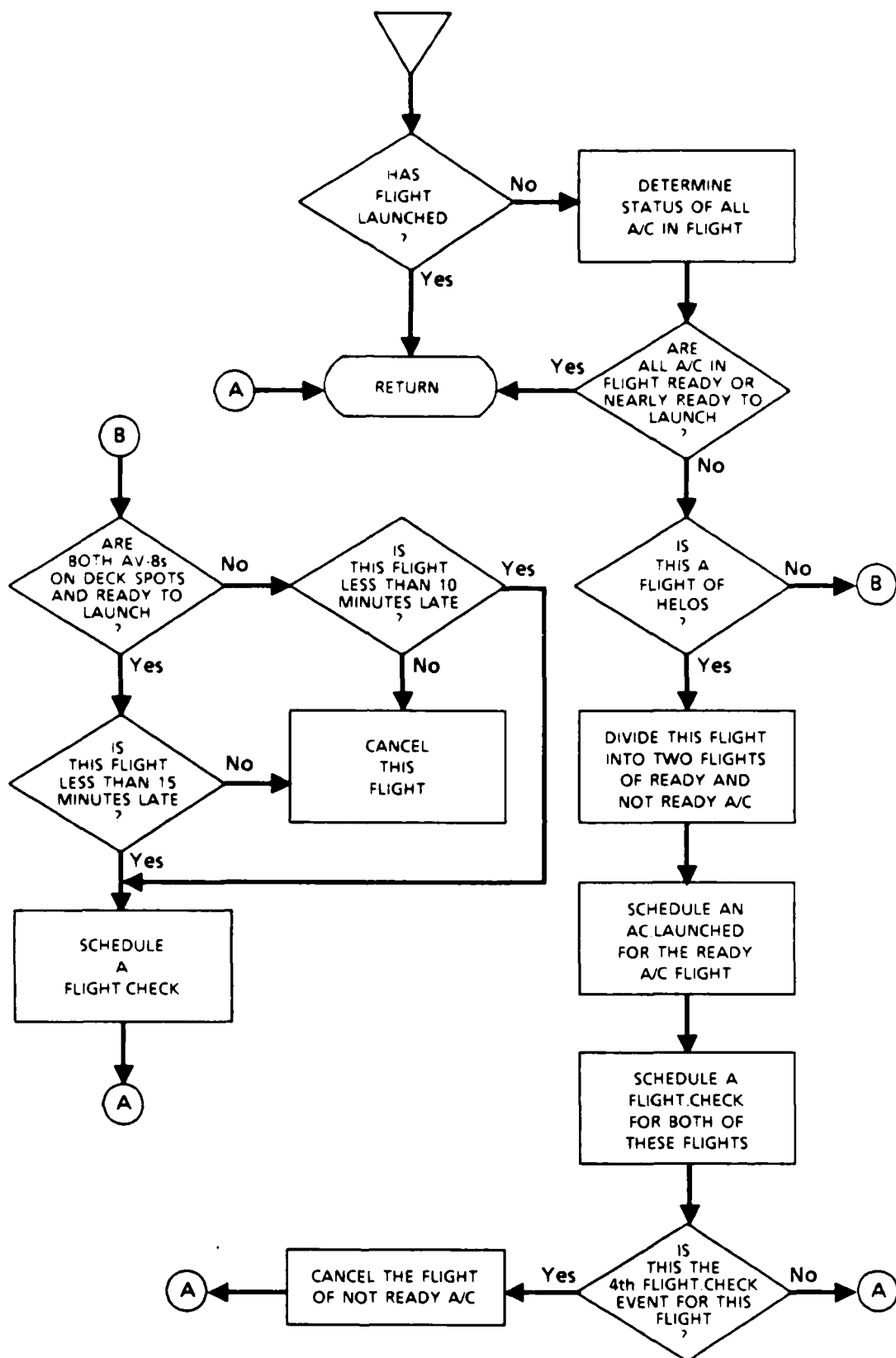
3.1.12 EVENT FLIGHT.CHECK GIVEN FLIGHT

The function of this event is to check whether or not a flight has launched, and if it has not, determine whether or not it should be cancelled. This event is scheduled in event DECK.DECISION, and occurs shortly after the scheduled launch time of FLIGHT.

If this is a flight of helicopters, the aircraft ready for takeoff will be launched at this point. If this is the fourth time that a FLIGHT.CHECK event has occurred for this FLIGHT, the aircraft that are not on deck in the final stages of preparation will have their launch times cancelled.

If this is a flight of AV-8s, both aircraft in the flight must be ready prior to launching the flight, because AV-8s always fly in groups of two. After a certain amount of time past the scheduled launch time, the flight will be cancelled.

If the flight has not been cancelled, another event FLIGHT.CHECK is scheduled.

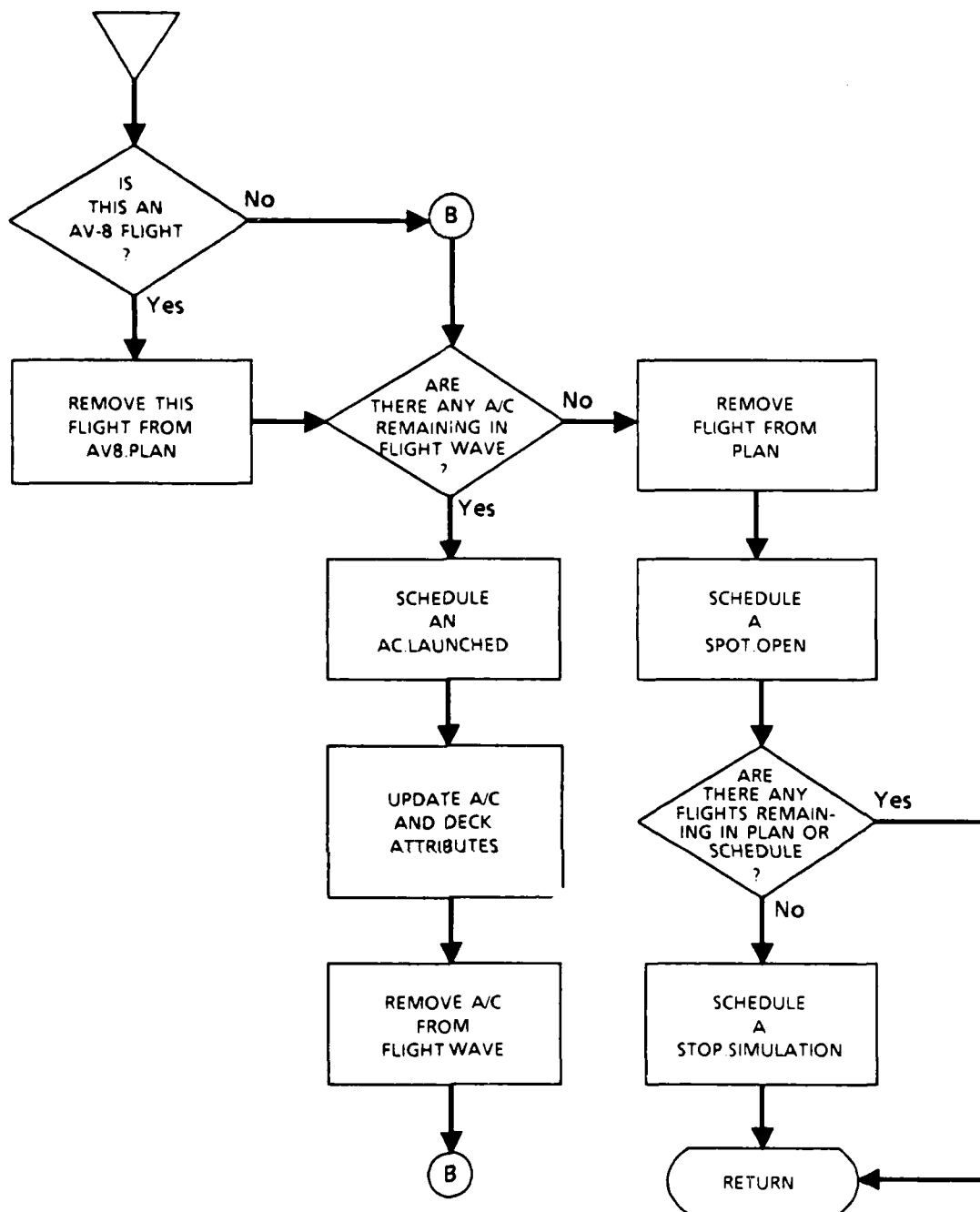


EVENT FLIGHT.CHECK GIVEN FLIGHT

3.1.13 EVENT FLIGHT.LAUNCH GIVEN FLIGHT

This event occurs when all aircraft scheduled to launch in this flight have completed refueling and loading, and are ready on deck spots to launch. Each aircraft in the flight has an event AC.LAUNCHED scheduled, and an event SPOT.OPEN is scheduled to follow the launch of the last aircraft in this flight.

If this was the last flight in the PLAN or SCHEDULE, an event STOP.SIMULATION is scheduled.



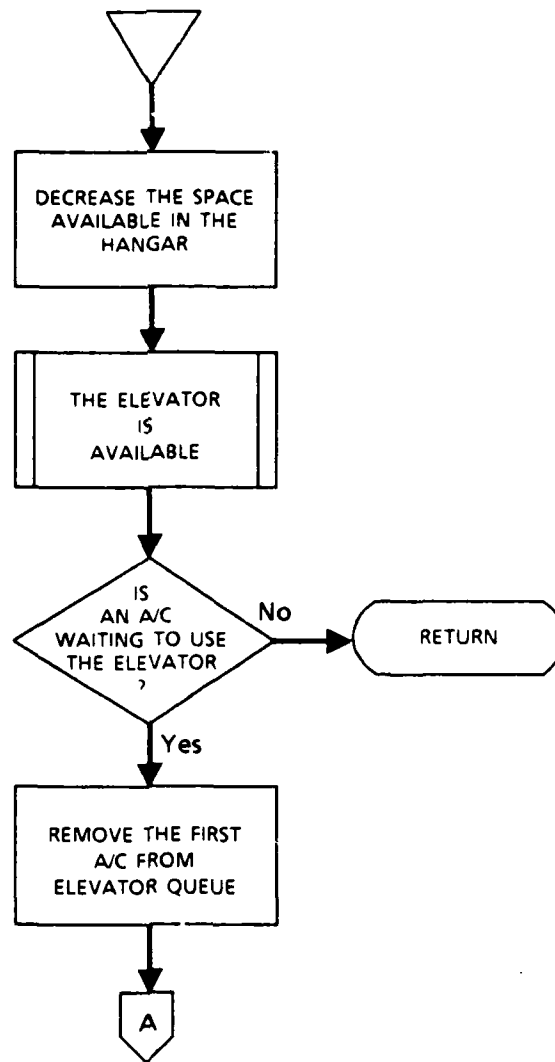
EVENT FLIGHT.LAUNCH GIVEN FLIGHT

3.1.14 EVENT HANGAR.ARRIVAL GIVEN AC

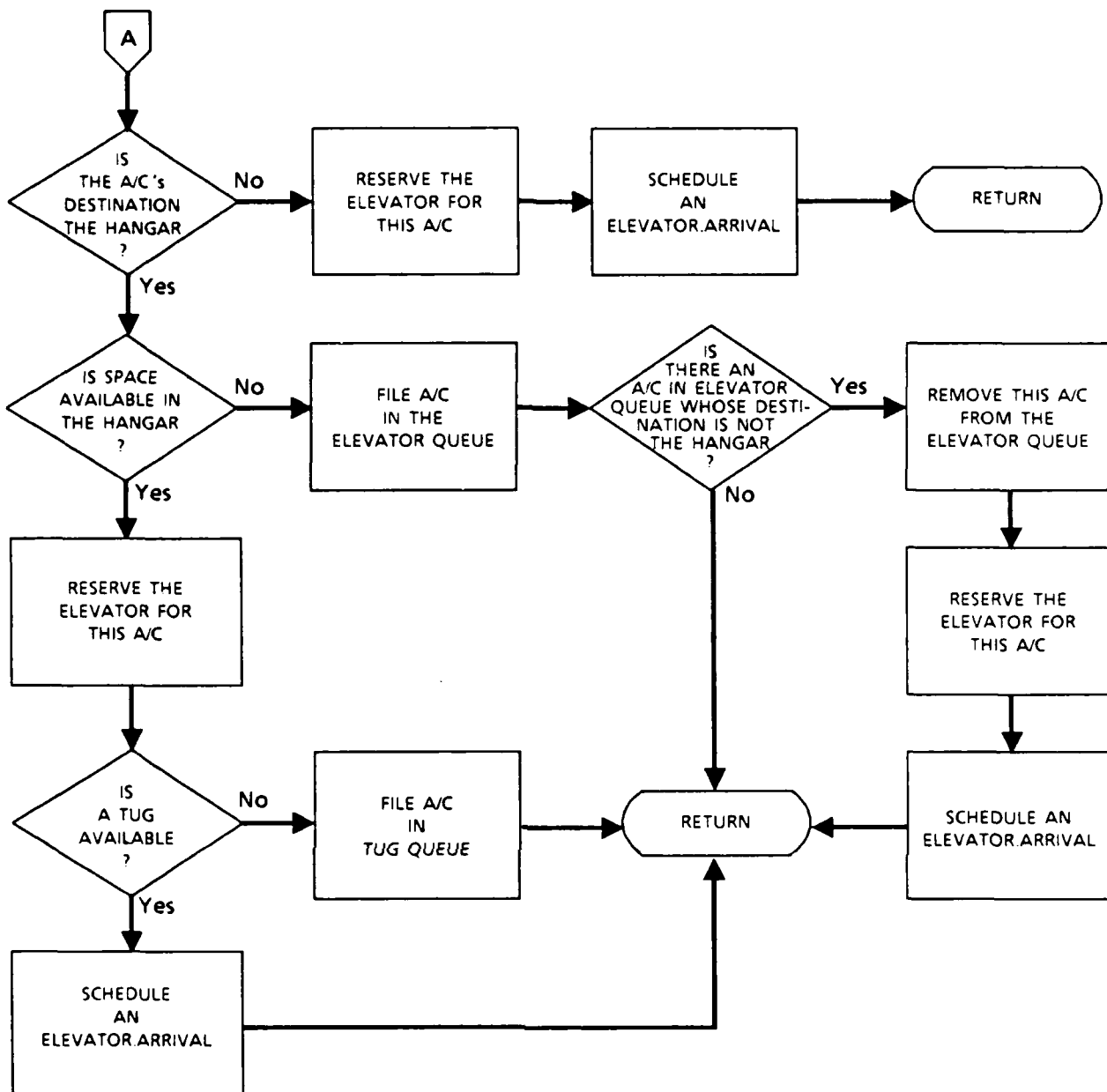
This event occurs when AC has arrived by elevator to the hangar deck. The hangar deck's tug is always available to remove aircraft from the elevator, so the elevator will soon become available, and the elevator queue is checked for waiting aircraft.

If an aircraft is waiting for the elevator, and its destination is the hangar, the hangar is checked to determine if there is space available. If these conditions are met, the elevator is reserved. If a tug is available, an ELEVATOR.ARRIVAL event is scheduled.

If an aircraft is waiting for the elevator, and its destination is the flight deck, the elevator is reserved and an ELEVATOR.ARRIVAL event is scheduled. (The tug on the hangar deck is assumed to be available whenever the elevator is.)



EVENT HANGAR.ARRIVAL GIVEN AC



CONTINUED: EVENT HANGAR.ARRIVAL

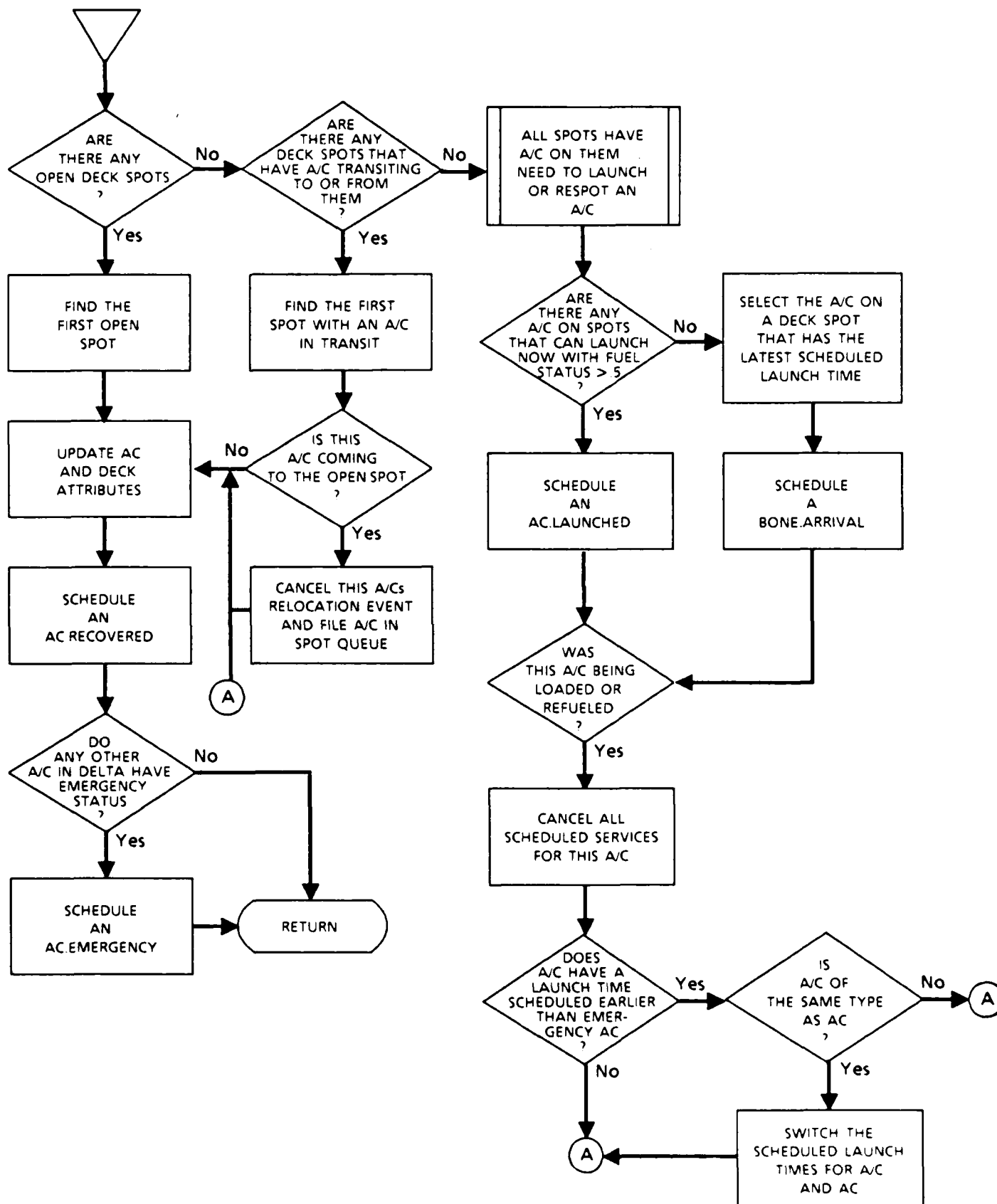
3.1.15 EVENT SPOT.EMERGENCY GIVEN AC

This event occurs when aircraft AC has entered a fuel emergency situation while waiting in the delta pattern to be assigned a deck spot for recovery to the ship.

If there are any deck spots open, the first open spot is reserved for AC and an AC.RECOVERED event is scheduled.

If there are no open spots, but there is a deck spot with an aircraft transiting to or from it, then this spot is reserved for AC, and an AC.RECOVERED event is scheduled. If the aircraft in transit was not leaving the spot, the relocation event for this aircraft is cancelled and the aircraft is filed in the spot queue to be considered for another spot.

If there are no open spots and no aircraft in transit to or from a spot, it is necessary to displace an aircraft to recover AC. The fuel status and launch time of each aircraft currently on a deck spot is checked to find an aircraft that has at least half a tank of fuel and is capable of launching immediately. If an aircraft is found meeting these requirements, an AC.LAUNCHED event is scheduled to clear a spot for AC to recover. If an aircraft cannot be found to launch, the aircraft on a deck spot with the latest launch time will schedule a BONE.ARRIVAL event. If the displaced aircraft was being refueled or loaded, the scheduled completion of these events is cancelled. If the emergency aircraft and the displaced aircraft are of the same type, and the displaced aircraft has an earlier scheduled launch time, these aircraft exchange launch times to minimize the potential for launch delay caused by the emergency recovery.



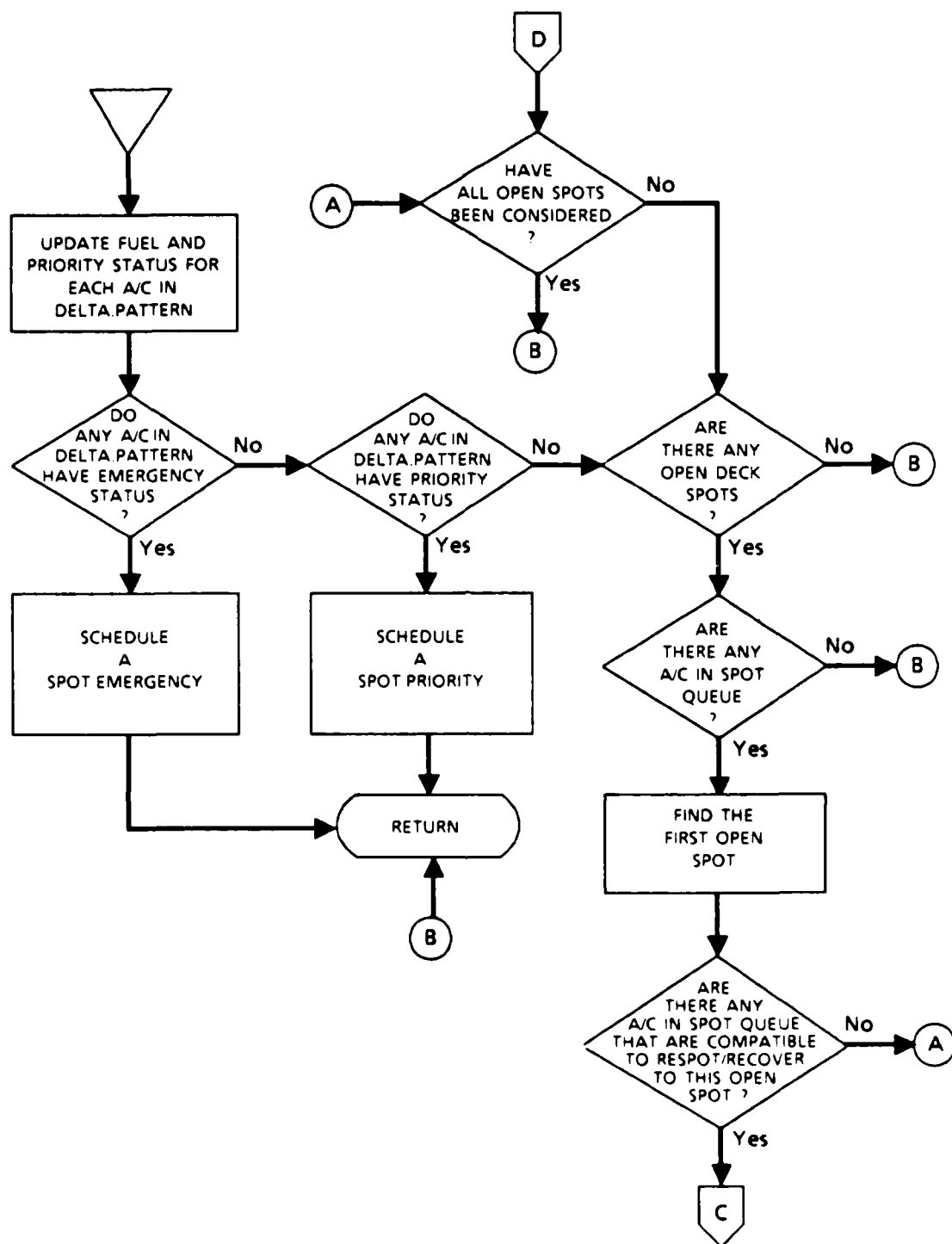
EVENT SPOT.EMERGENCY GIVEN AC

3.1.16 EVENT SPOT.OPEN

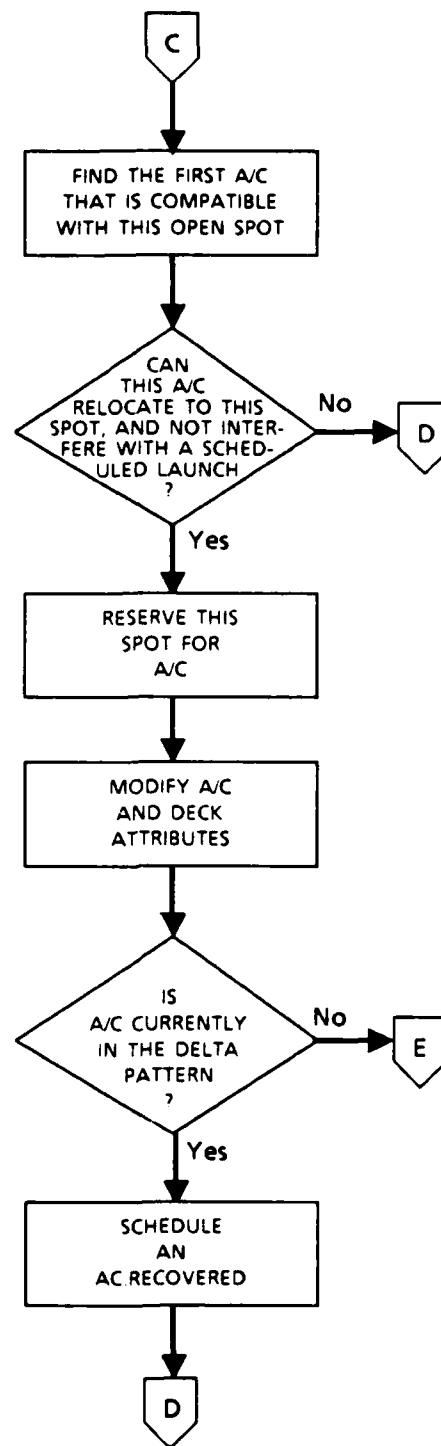
This event occurs at regular intervals (see event DELTA.UPDATE.CHK), and whenever a deck spot has been vacated by another aircraft. The fuel status of each aircraft in the delta pattern is updated and checked to see if it entitles the aircraft for priority or emergency treatment.

If an aircraft is qualified for special treatment, either event SPOT.EMERGENCY or SPOT.PRIORITY , as appropriate, is scheduled.

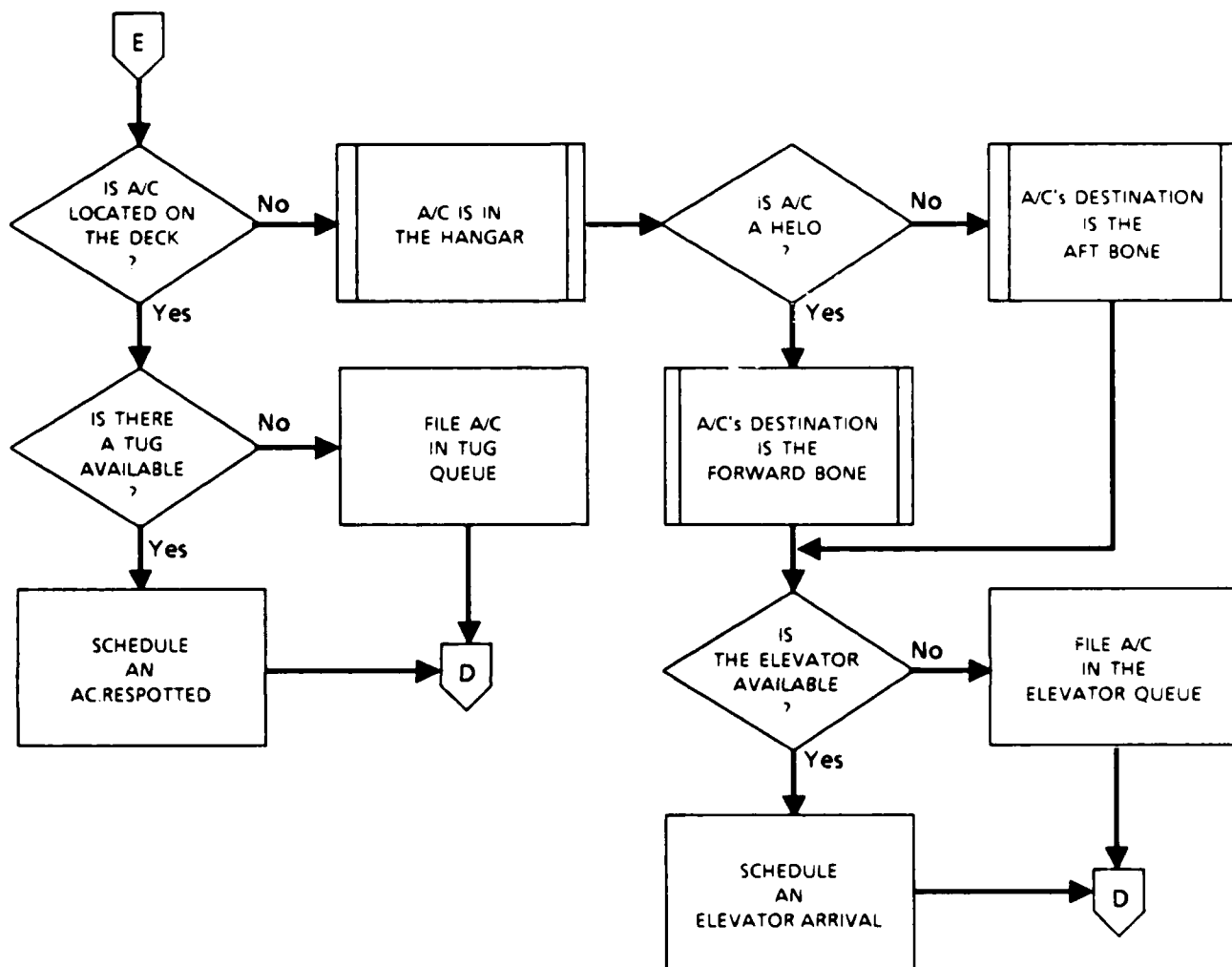
If no aircraft qualifies for special treatment, aircraft in the spot queue are considered for recovery or respotting to any open spots. Any aircraft that are compatible with an open spot, and will not interfere with another aircraft's launch are scheduled for an AC.RECOVERED or AC.RESPOTTED event.



EVENT SPOT.OPEN



(CONTINUED) EVENT SPOT.OPEN



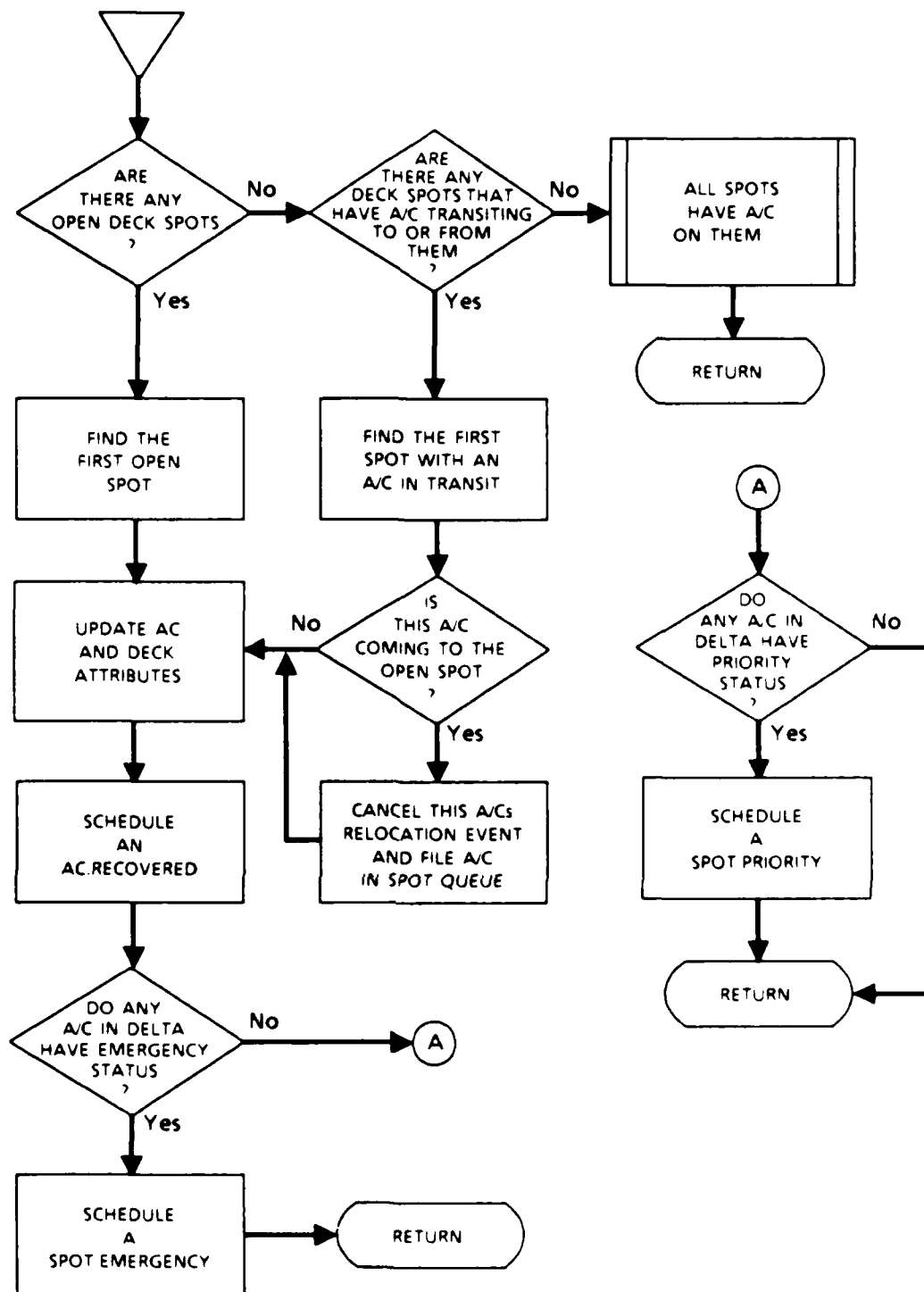
(CONTINUED) EVENT SPOT.OPEN

3.1.17 EVENT SPOT.PRIORITY GIVEN AC

This event occurs when aircraft AC has entered a fuel priority situation while waiting in the delta pattern to be assigned a deck spot for recovery to the ship.

If there are any deck spots open, the first open spot is reserved for AC and an AC.RECOVERED event is scheduled.

If there are no open spots, but there is a deck spot with an aircraft transiting to or from it, then this spot is reserved for AC, and an AC.RECOVERED event is scheduled. If the aircraft in transit was not leaving the spot, the relocation event for this aircraft is cancelled and the aircraft is filed in the spot queue to be considered for another spot.



EVENT SPOT.PRIORITY GIVEN AC

3.2 DESCRIPTION OF ENTITIES AND THEIR ATTRIBUTES

The model utilizes two entity structures: the permanent entity ACE and the temporary entity FLIGHTE. A copy of ACE is created for each aircraft that participates in the simulation. The attributes of ACE are defined as follows:

- :AC.DELTA.ARRIVAL.TIME - The time at which ACE arrived to the delta pattern
- :AC.DESTINATION - The next location planned for ACE
- :AC.FLYING.TIME - The amount of time that ACE can/could remain airborne with the fuel it currently has on board
- :AC.FUEL.STAT - The percent of a full fuel tank that ACE has on board
- :AC.ID - An identification number assigned to ACE
- :AC.LAUNCH.TIME - ACE's next scheduled launch time
- :AC.LOAD.STAT - The percent of a full load that ACE has on board
- :AC.LOCATION - The present location of ACE
- :AC.OP.STAT - The operational status of ACE
- :AC.PRIORITY - The priority assigned to ACE for recovery when in the delta pattern
- :AC.RECOVERY.TIME - The time at which ACE was recovered to the flight deck
- :AC.SERVICE.FLAG - Equals 1 if ACE is being loaded or refueled, 0 otherwise
- :AC.TAKEOFF.TIME - The time at which ACE launched from the ship
- :AC.TYPE - Equals 1 if ACE is a CH-46, 2 if a CH-53, and 3 if an AV-8

A FLIGHTE entity is created for each flight that is to be scheduled during the simulation. The attributes of FLIGHTE are defined as follows:

- :FLT.AC.NUM - The number of aircraft assigned to FLIGHTE
- :FLT.AC.RDY - The number of aircraft assigned to FLIGHTE that are ready to launch
- :FLT.AC.TYPE - The type of aircraft in this flight; corresponds with AC.TYPE
- :FLT.DELAY - The amount of time that FLIGHTE is late for takeoff
- :FLT.NUM - An identification number assigned to FLIGHTE

:FLT.TIME - The time that FLIGHTE is scheduled to launch
:FLT.WAVE - The set that contains the aircraft assigned to FLIGHTE

3.3 MEASURES OF EFFECTIVENESS

The model currently has a number of bookkeeping routines built in that generate a number of potential measures of effectiveness. Tables 1-3 list portions of the summary file that displays this output for a sample run.

The first table compares the planned flight schedule with the actual launches as they occurred. This information allows the user to investigate delays and cancelled missions by aircraft and flight. The second table displays in chronological order the times at which launches occurred by aircraft type. This information can be used to create histograms and compare launch rates. The third table lists a number of measures that track utilization, average queue size, and completion times for a number of service related activities and structures.

TABLE 1

COMPARISON OF THE PLANNED AND ACTUAL LAUNCH TIMES
(OUTPUT FOR 10 AV-8 CASE)

FLIGHT	1	SCHEDULED	AT	20	LAUNCHED	A/C	1	AT	14.3
FLIGHT	1	SCHEDULED	AT	20	LAUNCHED	A/C	2	AT	14.8
FLIGHT	1	SCHEDULED	AT	20	LAUNCHED	A/C	3	AT	15.2
FLIGHT	1	SCHEDULED	AT	20	LAUNCHED	A/C	4	AT	15.5
FLIGHT	2	SCHEDULED	AT	21	LAUNCHED	A/C	11	AT	16.8
FLIGHT	2	SCHEDULED	AT	21	LAUNCHED	A/C	12	AT	17.5
FLIGHT	3	SCHEDULED	AT	35	LAUNCHED	A/C	5	AT	31.1
FLIGHT	3	SCHEDULED	AT	35	LAUNCHED	A/C	6	AT	31.4
FLIGHT	3	SCHEDULED	AT	35	LAUNCHED	A/C	7	AT	31.9
FLIGHT	3	SCHEDULED	AT	35	LAUNCHED	A/C	8	AT	32.3
FLIGHT	4	SCHEDULED	AT	36	LAUNCHED	A/C	13	AT	29.5
FLIGHT	4	SCHEDULED	AT	36	LAUNCHED	A/C	14	AT	30.1
FLIGHT	5	SCHEDULED	AT	45	LAUNCHED	A/C	17	AT	43.9
FLIGHT	5	SCHEDULED	AT	45	LAUNCHED	A/C	18	AT	44.6
FLIGHT	6	SCHEDULED	AT	50	LAUNCHED	A/C	19	AT	50.0
FLIGHT	6	SCHEDULED	AT	50	LAUNCHED	A/C	20	AT	50.6
FLIGHT	7	SCHEDULED	AT	65	LAUNCHED	A/C	9	AT	60.7
FLIGHT	7	SCHEDULED	AT	65	LAUNCHED	A/C	10	AT	60.1
FLIGHT	8	SCHEDULED	AT	66	LAUNCHED	A/C	15	AT	63.8
FLIGHT	8	SCHEDULED	AT	66	LAUNCHED	A/C	16	AT	63.1
FLIGHT	9	SCHEDULED	AT	75	LAUNCHED	A/C	21	AT	71.7
FLIGHT	9	SCHEDULED	AT	75	LAUNCHED	A/C	22	AT	72.2
FLIGHT	26	SCHEDULED	AT	216	LAUNCHED	A/C	13	AT	233.1
FLIGHT	26	SCHEDULED	AT	216	LAUNCHED	A/C	14	AT	232.6
FLIGHT	27	SCHEDULED	AT	225	LAUNCHED	A/C	17	AT	225.5
FLIGHT	27	SCHEDULED	AT	225	LAUNCHED	A/C	18	AT	226.1
FLIGHT	28	SCHEDULED	AT	230	LAUNCHED	A/C	19	AT	0
FLIGHT	28	SCHEDULED	AT	230	LAUNCHED	A/C	20	AT	0
FLIGHT	29	SCHEDULED	AT	245	LAUNCHED	A/C	9	AT	253.1
FLIGHT	29	SCHEDULED	AT	245	LAUNCHED	A/C	10	AT	253.5
FLIGHT	30	SCHEDULED	AT	246	LAUNCHED	A/C	15	AT	264.7
FLIGHT	30	SCHEDULED	AT	246	LAUNCHED	A/C	16	AT	277.1
FLIGHT	31	SCHEDULED	AT	255	LAUNCHED	A/C	21	AT	259.9
FLIGHT	31	SCHEDULED	AT	255	LAUNCHED	A/C	22	AT	260.5
FLIGHT	32	SCHEDULED	AT	260	LAUNCHED	A/C	23	AT	0
FLIGHT	32	SCHEDULED	AT	260	LAUNCHED	A/C	24	AT	0
FLIGHT	42	SCHEDULED	AT	345	LAUNCHED	A/C	21	AT	345.9
FLIGHT	42	SCHEDULED	AT	345	LAUNCHED	A/C	22	AT	346.3
FLIGHT	43	SCHEDULED	AT	350	LAUNCHED	A/C	23	AT	351.0
FLIGHT	43	SCHEDULED	AT	350	LAUNCHED	A/C	24	AT	351.3

TABLE 2

HISTOGRAM DATA

AC.TYPE	-	1	LAUNCH NO.	1	LAUNCH.TIME	-	14.3
AC.TYPE	-	1	LAUNCH NO.	2	LAUNCH.TIME	-	14.8
AC.TYPE	-	1	LAUNCH NO.	3	LAUNCH.TIME	-	15.2
AC.TYPE	-	1	LAUNCH NO.	4	LAUNCH.TIME	-	15.5
AC.TYPE	-	1	LAUNCH NO.	5	LAUNCH.TIME	-	31.1
AC.TYPE	-	1	LAUNCH NO.	6	LAUNCH.TIME	-	31.4
AC.TYPE	-	1	LAUNCH NO.	7	LAUNCH.TIME	-	31.9
AC.TYPE	-	1	LAUNCH NO.	8	LAUNCH.TIME	-	32.3
AC.TYPE	-	1	LAUNCH NO.	9	LAUNCH.TIME	-	60.1
AC.TYPE	-	1	LAUNCH NO.	10	LAUNCH.TIME	-	60.7
AC.TYPE	-	2	LAUNCH NO.	1	LAUNCH.TIME	-	16.8
AC.TYPE	-	2	LAUNCH NO.	2	LAUNCH.TIME	-	17.5
AC.TYPE	-	2	LAUNCH NO.	3	LAUNCH.TIME	-	29.5
AC.TYPE	-	2	LAUNCH NO.	4	LAUNCH.TIME	-	30.1
AC.TYPE	-	2	LAUNCH NO.	5	LAUNCH.TIME	-	63.1
AC.TYPE	-	2	LAUNCH NO.	6	LAUNCH.TIME	-	63.8
AC.TYPE	-	2	LAUNCH NO.	7	LAUNCH.TIME	-	95.6
AC.TYPE	-	2	LAUNCH NO.	8	LAUNCH.TIME	-	96.1
AC.TYPE	-	2	LAUNCH NO.	9	LAUNCH.TIME	-	121.8
AC.TYPE	-	2	LAUNCH NO.	10	LAUNCH.TIME	-	122.1
AC.TYPE	-	3	LAUNCH NO.	1	LAUNCH.TIME	-	43.9
AC.TYPE	-	3	LAUNCH NO.	2	LAUNCH.TIME	-	44.6
AC.TYPE	-	3	LAUNCH NO.	3	LAUNCH.TIME	-	50.0
AC.TYPE	-	3	LAUNCH NO.	4	LAUNCH.TIME	-	50.6
AC.TYPE	-	3	LAUNCH NO.	5	LAUNCH.TIME	-	71.7
AC.TYPE	-	3	LAUNCH NO.	6	LAUNCH.TIME	-	72.2
AC.TYPE	-	3	LAUNCH NO.	7	LAUNCH.TIME	-	77.2
AC.TYPE	-	3	LAUNCH NO.	8	LAUNCH.TIME	-	77.7
AC.TYPE	-	3	LAUNCH NO.	9	LAUNCH.TIME	-	108.0
AC.TYPE	-	3	LAUNCH NO.	10	LAUNCH.TIME	-	108.4
AC.TYPE	-	3	LAUNCH NO.	27	LAUNCH.TIME	-	329.6
AC.TYPE	-	3	LAUNCH NO.	28	LAUNCH.TIME	-	330.2
AC.TYPE	-	3	LAUNCH NO.	29	LAUNCH.TIME	-	345.9
AC.TYPE	-	3	LAUNCH NO.	30	LAUNCH.TIME	-	346.9
AC.TYPE	-	3	LAUNCH NO.	31	LAUNCH.TIME	-	351.0
AC.TYPE	-	3	LAUNCH NO.	32	LAUNCH.TIME	-	351.3

TABLE 3

MEASURES OF EFFECTIVENESS

	<u>Average</u>	<u>Variant</u>	<u>Minimum</u>	<u>Maximum</u>
SPOT.Q	4.21	7.97		11.00
REFUEL.Q	0.01	0.01		2.00
REFUELER	0.82	1.19		4.00
N.DELTA.PATTERN	4.07	8.17		10.00
N.BONE.FWD	1.68	4.86		6.00
N.BONE.AFT	6.10	3.44		10.00
BONE.TOTAL	8.37	8.57		12.00
N.HANGAR	2.27	15.37		13.00
TUG	0.39	0.59		4.00
N.TUG.Q	0.01	0.01		2.00
LOADER	1.33	0.74		2.00
N.LOADER.Q	0.69	0.84		3.00
LAUNCH.DELAY	6.04	80.95		31.11
HELO.MISSION.LENGTH	57.78	11.73	51.22	67.13
AV8.MISSION.LENGTH	29.86	20.87	23.70	40.48
HELO.FLYING.TIME	79.71	136.27	55.61	117.09
AV8.FLYING.TIME	45.42	39.60	29.38	55.61
HELO.RECOVER.TIME	21.93	144.35	3.38	56.94
AV8.RECOVER.TIME	15.55	49.68	3.69	30.49
TTLOAD	9.01	69.72	1.23	28.36
TTRESPOT	3.43	1.36	0.47	5.83
TIRECOVER	3.49	0.75	1.82	6.45
TTARRIV.ELEVATOR	6.56	2.33	3.59	8.94
TTARRIV.DECK	1.52	0.47	0.80	2.85
TTARRIV.HANGAR	1.32	0.32	0.62	2.81
TTUNLOAD	16.68	14.79	10.20	27.14

NO. EMERG.RECOVERIES - 0

NO. PRIORITY.RECOVERIES - 14

NO. STD.RECOVERIES - 102

NO. LAUNCHES - 96

NO. REPLACED.AC - 0

NO. CANCELLED.MISSIONS - 6

NO. FLIGHTS STILL IN SCHEDULE - 0 STILL IN PLAN - 0

Chapter 4

ANALYSIS OF THE AV-8/HELO MIX PROBLEM

The inputs that were used to run the model for this problem are listed in table 4. The distribution inputs are based on exercise results, however, all have been modified to account for expedients that would be taken during the conduct of an actual assault. These modifications ruled out using distributions which fit the exercise data, but provided approximations for the minimum, maximum, mean, and mode. In several cases, a shifted Beta distribution was chosen, and the parameters were determined by these approximations. Table 5 presents the distributions with associated parameters used for the event scheduling in model runs, and the random number stream assignments. In the absence of actual assault data, goodness-of-fit tests were not conducted.

The simulation begins with the aircraft at locations and statuses as shown in table 6. Helicopters of the second wave are in the delta pattern with full fuel tanks to minimize the interval between the first and second wave; it takes less time to recover-load-launch than it does to respot-start-load-launch. The first flight of AV-8s launches after the second wave. Thereafter, an AV-8 flight follows every wave if an AV-8 flight is available. The schedule of launches for each case is shown in table 7.

The primary MOE for this problem is the force buildup rate ashore for each of the aircraft mixes. The model tracks the buildup rate by recording the time that each aircraft launches; this removes from consideration any

TABLE 4
MODEL INPUTS FOR AIRCRAFT TYPES

	<u>CH-46</u>	<u>CH-53</u>	<u>AV-8</u>
AVERAGE FLYING SPEED (m/s)	65	65	200
AVERAGE FLYING TIME	120	150	75
FUEL CAPACITY (lb)	15,000	25,000	15,000
AVERAGE TIME TO REFUEL	5	6	5
AVERAGE TIME TO LOAD	5	10	16
STD DEV OF TIME TO LOAD	--	--	2
AVERAGE TIME TO RECOVER	4	4	3
STD DEV OF TIME TO RECOVER	.5	.5	.5
AVERAGE TIME TO RESPOT	4	4	3
STD DEV OF TIME TO RESPOT	1	1	1
AVERAGE TIME TO MOVE TO ELEVATOR	6	6	6
STD DEV OF TIME TO MOVE TO ELEVATOR	1.5	1.5	1.5
AVERAGE TIME TO COMPLETE MISSION	16	16	16
EMERGENCY FUEL LEVEL (%)	17	15	25
PRIORITY FUEL LEVEL (%)	25	20	35
PREFERRED LAUNCH SPOTS	1,2,3,4	5,6	1,2
PREFERRED RECOVERY SPOTS	1,2,3,4	5,6	1,2,5,6

NOTE: All times are in minutes.

TABLE 5

Event	Distribution	Stream
AC.LAUNCHED	UNIFORM (.5, 1) + INTERVAL FOR OTHER LAUNCHES	5
AC.LOADED	NORMAL... SEE TABLE 4 FOR PARAMETERS SPECIFIC TO AIRCRAFT TYPE	3
AC.RECOVERED	NORMAL... SEE TABLE 4	1
AC.REFUELED	FUNCTION OF FUEL STATUS AND TIME TO REFUEL PLUS DELAY NORMAL (1., .25)	4
AC.RESPOTTED	NORMAL... SEE TABLE 4	2
BONE.ARRIVAL	NORMAL... SAME AS AC.RESPOTTED	2
DECK.ARRIVAL	BETA (1.5, 5) MIN = .5 MAX = 4.5 MEAN = 1.4 MODE = .9	7
DELTA.ARRIVAL	FUNCTION OF DISTANCE TO SHORE AND A/C SPEED PLUS TIME TO UNLOAD BETA (1.5, 3)	6
ELEVATOR.ARRIVAL	NORMAL (6, 1.5)	8
FLIGHT.LAUNCH	UNIFORM (.1, .5)	9
FLIGHT.CHECK	BETA (1.5, 3) MIN = 1 MAX = 3 MEAN = 1.7 MODE = 1.4	9
HANGAR.ARRIVAL	BETA (1.5, 5) MIN = .5 MAX = 4.5 MEAN = 1.4 MODE = .9	7

TABLE 6
INITIAL AIRCRAFT LOCATIONS AND STATUSES

<u>Aircraft number</u>	<u>Aircraft type</u>	<u>Location</u>	<u>Fuel status</u>	<u>Load status</u>
1	CH-46	FWD BONE	1.0	.0
2	CH-46	FWD BONE	1.0	.0
3	CH-46	FWD BONE	1.0	.0
4	CH-46	FWD BONE	1.0	.0
5	CH-46	DELTA	1.0	.0
6	CH-46	DELTA	1.0	.0
7	CH-46	DELTA	1.0	.0
8	CH-46	DELTA	1.0	.0
9	CH-46	FWD BONE	1.0	.0
10	CH-53	FWD BONE	1.0	.0
11	CH-53	AFT BONE	1.0	.0
12	CH-53	AFT BONE	1.0	.0
13	CH-53	DELTA	1.0	.0
14	CH-53	DELTA	1.0	.0
15	CH-53	AFT BONE	1.0	.0
16	CH-53	AFT BONE	1.0	.0
17	AV-8	AFT BONE	1.0	1.0
18	AV-8	AFT BONE	1.0	1.0
19	AV-8	AFT BONE	1.0	1.0
20	AV-8	AFT BONE	1.0	1.0
21	AV-8	HANGAR	.0	.0
22	AV-8	HANGAR	.0	.0
23	AV-8	HANGAR	.0	.0
24	AV-8	HANGAR	.0	.0
25	AV-8	HANGAR	.0	.0
26	AV-8	HANGAR	.0	.0
27	AV-8	HANGAR	.0	.0
28	AV-8	HANGAR	.0	.0

TABLE 7
FLIGHT SCHEDULE FOR 6, 8, 10, AND 12 AV-8 CASES

WAVE NUMBER	LAUNCH TIME	AIRCRAFT TYPE	NUMBER OF AIRCRAFT	NUMBER OF AV-8's			
				12	10	8	6
1	20	CH-46	4				
1	21	CH-53	2				
2	35	CH-46	4				
2	36	CH-53	2				
	45	AV-8	2	X	X	X	X
	50	AV-8	2	X	X	X	
3	65	CH-46	2				
3	66	CH-53	2				
	75	AV-8	2	X	X	X	X
	80	AV-8	2	X	X		
4	95	CH-46	4				
4	96	CH-53	2				
	105	AV-8	2	X	X	X	X
	110	AV-8	2	X			
5	125	CH-46	4				
5	126	CH-53	2				
	135	AV-8	2	X	X	X	X
	140	AV-8	2	X	X	X	
6	155	CH-46	2				
6	156	CH-53	2				
	165	AV-8	2	X	X	X	X
	170	AV-8	2	X	X		
7	185	CH-46	4				
7	186	CH-53	2				
	195	AV-8	2	X	X	X	X
	200	AV-8	2	X			
8	215	CH-46	4				
8	216	CH-53	2				
	225	AV-8	2	X	X	X	X
	230	AV-8	2	X	X	X	
9	245	CH-46	2				
9	246	CH-53	2				
	255	AV-8	2	X	X	X	X
	260	AV-8	2	X	X		
10	275	CH-46	4				
10	276	CH-53	2				
	285	AV-8	2	X	X	X	X
	290	AV-8	2	X			
11	305	CH-46	4				
11	306	CH-53	2				
	315	AV-8	2	X	X	X	X
	320	AV-8	2	X	X	X	
12	335	CH-46	2				
12	336	CH-53	2				
	345	AV-8	2	X	X	X	X
	350	AV-8	2	X	X		
NUMBER OF AV-8 FLIGHTS:				22	19	15	11

NOTE: X signifies that the AV-8 flight is scheduled for that AV-8 case.

factors that might affect the transit from ship to shore that are not resident in the flight deck operations. In other words, the best case is the case that is able to launch the most aircraft in the shortest time. Three other MOEs that are related to buildup are the average launch delay, the number of cancelled launches, and the number of AV-8 launches. Table 8 presents summary statistics for each case after ten replicates.

In order to introduce positive correlation between the cases, the method of common random numbers was employed as a variance reduction technique. The ten replicates for the twelve AV-8 case were run and the initial seeds for each random number stream were saved for each replicate. The ten replicates were then run for the other cases using the same initial seeds for each replicate.

The addition of two AV-8s from six to eight provides potentially eight more AV-8 sorties, with an average of 7.8 more sorties realized. The penalty paid for this increase is a reduction in the amount of time the average launch precedes its scheduled time by approximately 80 seconds. As the average launch is still ahead of schedule, a subjective judgement is made that the gain in AV-8 sorties outweighs the loss in timeliness, and the discussion continues with the eight AV-8 case designated as more effective than the case with six AV-8s.

For the cases of ten and twelve AV-8s, adding additional AV-8s appears to actually reduce the expected number of AV-8 launches. The increase in the number of possible AV-8 launches is almost identical to the increase in the number of missions cancelled between the eight, ten, and twelve AV-8 cases. Table 9 shows the results of a closer look at the cases of eight and ten AV-8s using paired differences and paired-t confidence intervals.

The results support the earlier observations, with 95 percent confidence, the expected number of additional missions cancelled lies in the interval (7.3, 11.3), and the expected number of additional AV-8 sorties lies in the interval (-2.8, 0.8). Also, the increase in launch delay is significant, and with 95 percent confidence the average launch will occur more than 5 minutes late for the 10 AV-8 case. Applying the Bonferroni inequality, the overall confidence that these three intervals simultaneously contain their respective true measures is at least 85 percent.

Figures 4-6 present graphical representations of the differences in buildup rate for the 6, 8, and 10 AV-8 cases. As can be seen, there is essentially no change in the buildup rate between the 6 and 8 AV-8 cases for helicopters, but there is a marked and consistent increase in the buildup rate for AV-8s. Between the 8 and 10 AV-8 cases, some decrease in the helicopter buildup

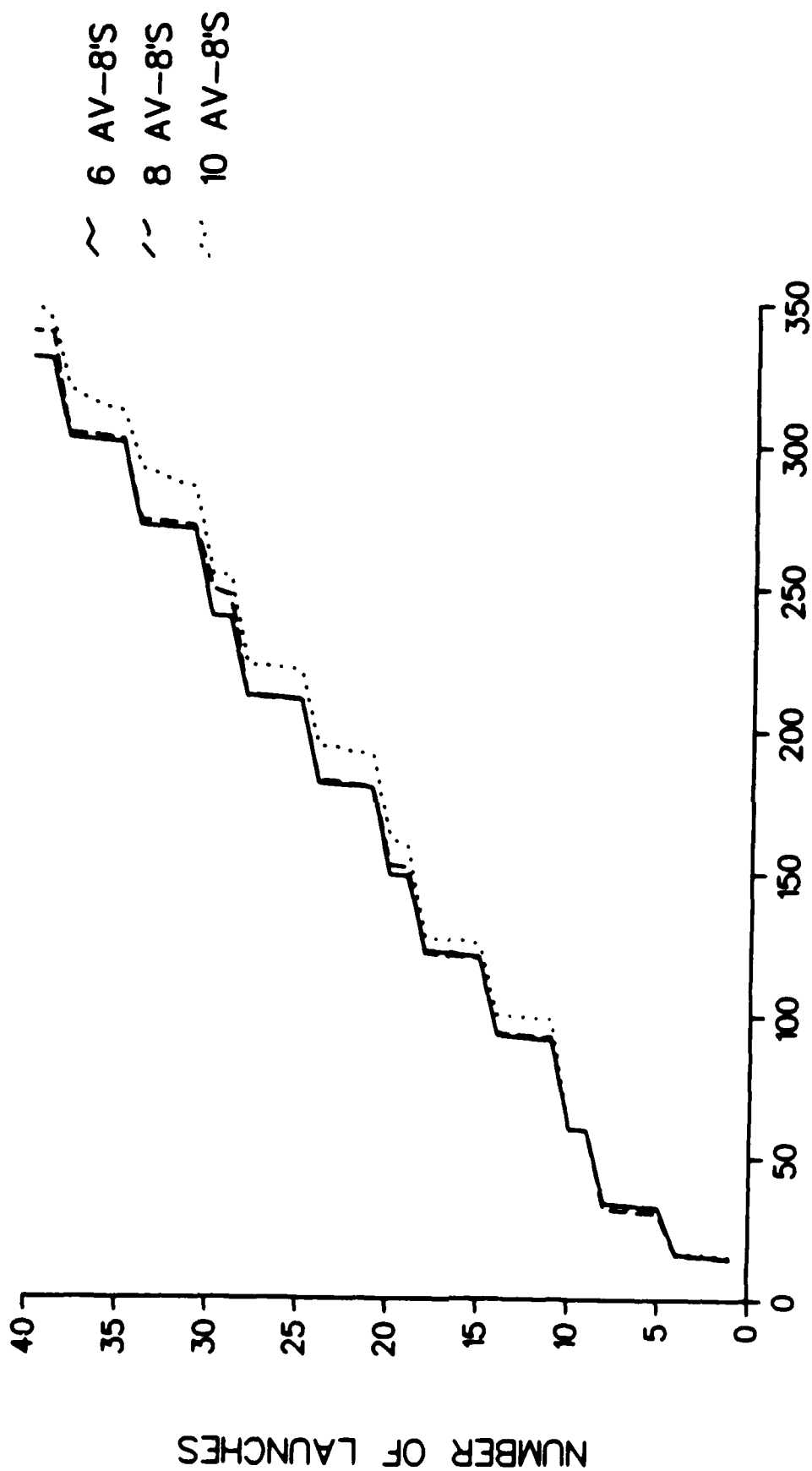
TABLE 8
SELECTED RESULTS AFTER TEN REPLICATES

<u>NUMBER OF AV-8's</u>	<u>AVERAGE NUMBER MISSIONS CANCELLED</u>	<u>AVERAGE TAKEOFF DELAY</u>	<u>AVERAGE NUMBER AV-8 LAUNCHES</u>	<u>POSSIBLE NUMBER AV-8 LAUNCHES</u>
6	0.0	-3.15	22.0	22
8	0.2	-1.73	29.8	30
10	9.5	4.67	28.8	38
12	16.1	6.38	28.6	44

TABLE 9

PAIRED-DIFFERENCE ANALYSIS FOR THE 8 AND 10 AV-8 CASES

Replicate	Number of missions cancelled			Average takeoff delay			Number of AV-8 launches			
	10 AV-8s	8 AV-8s	10 - 8	10 AV-8s	8 AV-8s	10 - 8	10 AV-8s	8 AV-8s	10 - 8	
1	6	0	6	6.0	-1.4	7.4	32	30	2	
2	13	2	11	4.4	-1.5	5.9	26	28	-2	
3	12	0	12	5.4	-2.4	7.8	26	30	-2	
4	10	0	10	4.4	-1.1	5.5	28	30	-2	
5	4	0	4	4.5	-3.4	7.9	34	30	4	
6	10	0	10	5.4	-1.6	7.0	28	30	-2	
7	12	0	12	5.9	-1.6	7.5	28	30	-2	
8	8	0	8	5.5	-0.6	6.1	30	30	0	
9	12	0	12	2.1	-0.8	2.9	26	30	-4	
10	8	0	8	3.1	-2.9	6.0	30	30	0	
$\bar{X} =$			9.3	$\bar{X} =$			6.4			-1.0
$S =$			2.75	$S =$			1.50			2.54
$d =$			1.97	$d =$			1.08			1.82
$r =$			0.21	$r =$			0.17			1.82
$CI =$			(7.3, 11.3)	$CI =$			(5.3, 7.5)			(-2.8, 0.8)



CH-46 LAUNCH TIMES

FIG. 4: BUILDUP RATE DURING A SIX-HOUR ASSAULT
 (CH-46)

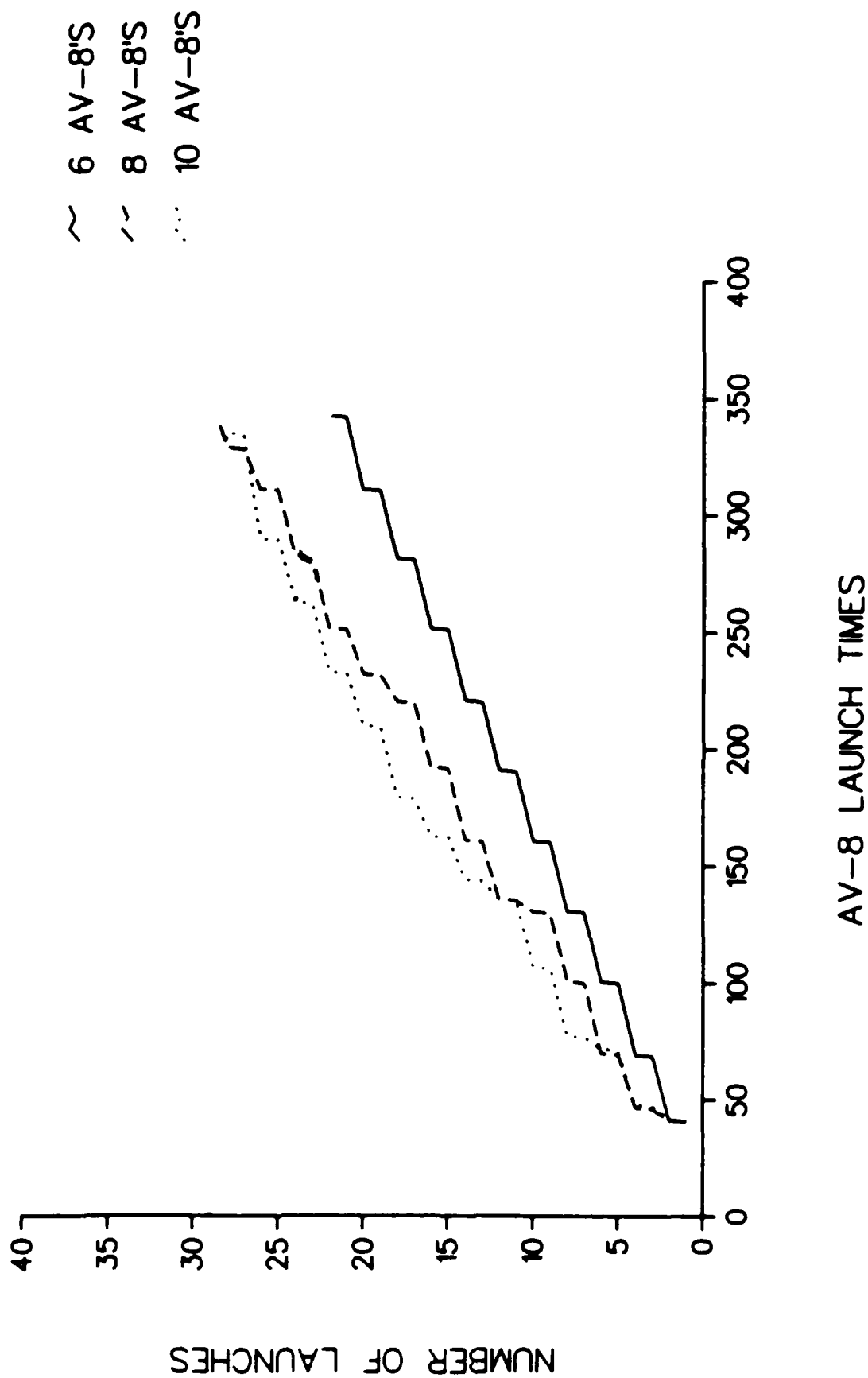


FIG. 5: BUILDUP RATE DURING A SIX-HOUR ASSAULT
(AV-8)

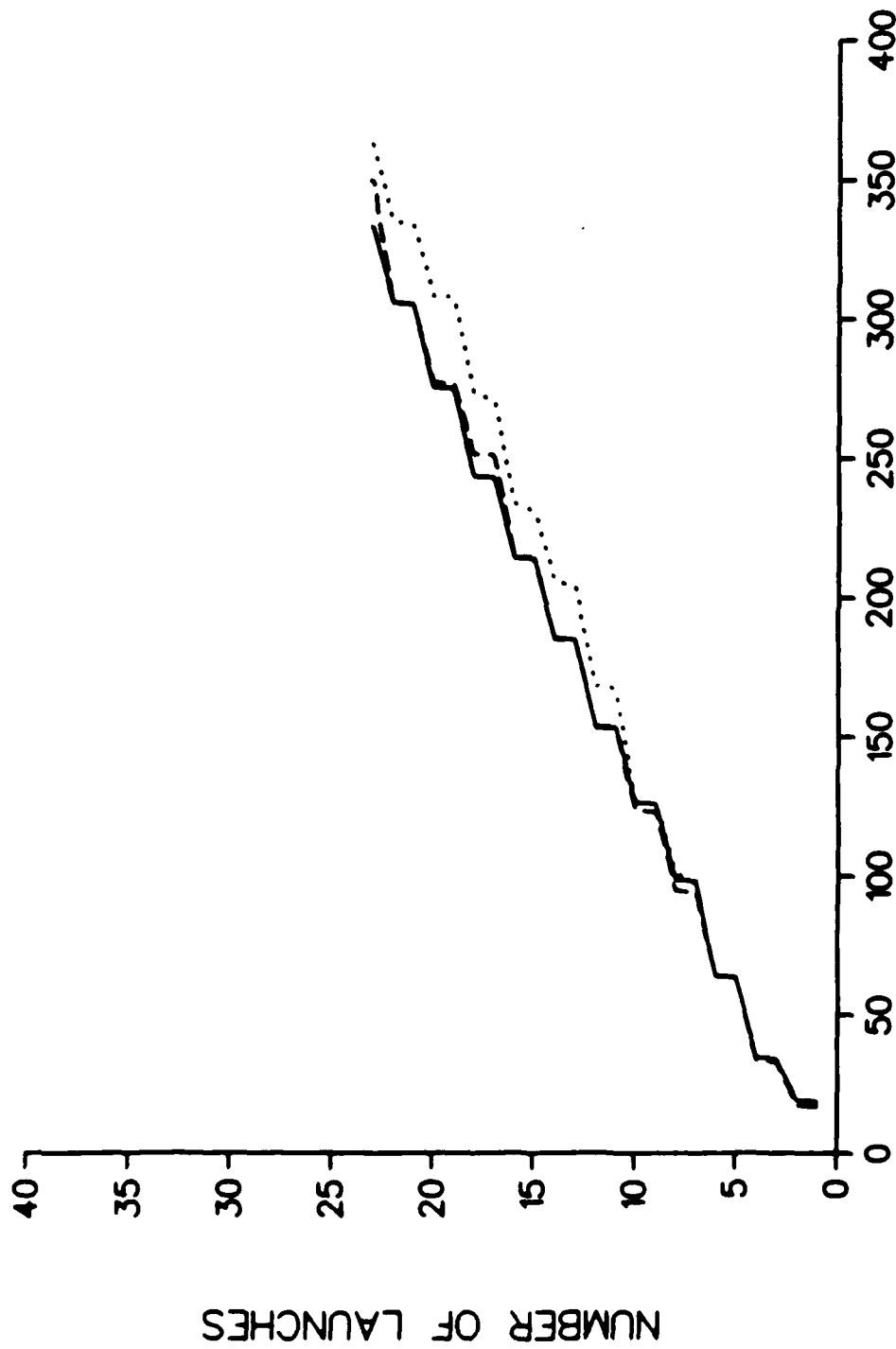


FIG. 6: BUILDUP RATE DURING A SIX-HOUR ASSAULT
 (CH-53)

rate is seen, and while the AV-8 buildup rate for the 10 AV-8 case appears to be generally above that of the 8 AV-8 case, the frequent intersections of the two buildup lines imply that there is no real improvement offered by 10 AV-8s in the long run.

Given the inputs, initial conditions, and launch schedule presented in this section, the optimum number of AV-8s for simultaneous flight deck operations with a composite helicopter squadron during an assault from an LHA is eight. It should be noted again that the unavailability of AV-8s due to combat attrition and reliability failure has not been modelled. If these are relevant factors, then the MAU should deploy with additional AV-8s in order to sustain operations at the level of eight.

APPENDIX A

AV-8/HELO MIX SIMULATION

PREAMBLE

EVENT NOTICES INCLUDE AC.LAUNCHED, AC.RECOVERED, AC.RESPOTTED, AC.LOADED,
AC.REFUELED, BONE.ARRIVAL, DELTA.ARRIVAL, DECK.DECISION,
SPOT.OPEN, FLIGHT.LAUNCH, STOP.SIMULATION, DELTA.UPDATE.CHK,
SPOT.EMERGENCY, SPOT.PRIORITY, HANGER.ARRIVAL,
ELEVATOR.ARRIVAL, DECK.ARRIVAL, FLIGHT.CHECK

EVERY AC.LAUNCHED HAS AN AC1 AND AN F1
EVERY AC.RECOVERED HAS AN AC2
EVERY AC.RESPOTTED HAS AN AC3
EVERY AC.LOADED HAS AN AC4
EVERY AC.REFUELED HAS AN AC5
EVERY BONE.ARRIVAL HAS AN AC6
EVERY DELTA.ARRIVAL HAS AN AC7
EVERY DECK.DECISION HAS A FLIGHT1
EVERY FLIGHT.LAUNCH HAS A FLIGHT2
EVERY SPOT.EMERGENCY HAS AN AC8
EVERY SPOT.PRIORITY HAS AN AC9
EVERY HANGER.ARRIVAL HAS AN AC10
EVERY ELEVATOR.ARRIVAL HAS AN AC11
EVERY DECK.ARRIVAL HAS AN AC12
EVERY FLIGHT.CHECK HAS A FLIGHT3

PERMANENT ENTITIES

EVERY ACE HAS AN AC.ID, AN AC.TYPE, AN AC.LOCATION, AN AC.FUEL.STAT,
AN AC.LOAD.STAT, AN AC.OP.STAT, AN AC.LAUNCH.TIME,
AN AC.DELTA.ARRIVAL.TIME, AN AC.PRIORITY, AN AC.DESTINATION,
AN AC.FLYING.TIME, AN AC.SERVICE.FLAG, AN AC.TAKEOFF.TIME,
AND AN AC.RECOVERY.TIME

AND MAY BELONG TO THE SHIP
AND MAY BELONG TO THE HANGER.DECK
AND MAY BELONG TO THE BONE.FWD
AND MAY BELONG TO THE BONE.AFT
AND MAY BELONG TO THE DELTA.PATTERN
AND MAY BELONG TO THE SET.TEMP
AND MAY BELONG TO THE REFUELER.Q
AND MAY BELONG TO THE LOADER.Q **AV-9'S ONLY
AND MAY BELONG TO THE LOAD.SET **AV-9'S ONLY
AND MAY BELONG TO THE TUG.Q
AND MAY BELONG TO THE FLT.WAVE
AND MAY BELONG TO THE SPOT.Q
AND MAY BELONG TO THE ELEVATOR.Q
AND MAY BELONG TO THE AC.RDY.SET
AND MAY BELONG TO THE AC.PRE.RDY.SET
AND MAY BELONG TO THE AC.NOT.RDY.SET

AND MAY BELONG TO THE ARRAY SPOT (THIS IS NOT A FUNCTIONAL LINE)

TEMPORARY ENTITIES

EVERY FLIGHT HAS A FLT.TIME, A FLT.AC.TYPE, A FLT.AC.NUM, A FLT.NUM,
A FLT.AC.RDY, AND A FLT.DELAY
AND OWNS A FLT.WAVE
AND MAY BELONG TO THE SCHEDULE
AND MAY BELONG TO THE PLAN
AND MAY BELONG TO THE AV8.PLAN

AFTER FILING IN FLT.WAVE CALL CHECK2
BEFORE FILING IN SCHEDULE CALL CHECK1
AFTER FILING IN SCHEDULE CALL CHECK1

DEFINE MINUTES TO MEAN UNITS
DEFINE MINUTE TO MEAN UNITS

DEFINE HANGER.DECK AS A SET RANKED BY AC.OP.STAT
DEFINE BONE.FWD AS A SET RANKED BY AC.LAUNCH.TIME
DEFINE BONE.AFT AS A SET RANKED BY AC.LAUNCH.TIME
DEFINE REFUELER.Q AS A SET RANKED BY LOW AC.LAUNCH.TIME, THEN BY AC.LOCATION
DEFINE LOADER.Q AS A SET RANKED BY LOW AC.LAUNCH.TIME **AV8'S
DEFINE LOAD.SET AS A SET **AV8'S
DEFINE TUG.Q AS A SET RANKED BY LOW AC.LAUNCH.TIME
DEFINE DELTA.PATTERN AS A SET RANKED BY LOW AC.FLYING.TIME
DEFINE SET.TEMP AS A SET
DEFINE ELEVATOR.Q AS A SET RANKED BY LOW AC.LOCATION,
THEN BY LOW AC.LAUNCH.TIME
DEFINE FLT.WAVE AS A SET RANKED BY LOW AC.LOCATION WITHOUT M ATTRIBUTE
DEFINE SPOT.Q AS A SET RANKED BY LOW AC.LAUNCH.TIME,
THEN BY LOW AC.FLYING.TIME
DEFINE AC.RDY.SET AS A SET RANKED BY LOW AC.LOCATION **WITHOUT M ATTRIBUTE
DEFINE AC.PRE.RDY.SET AS A SET RANKED BY LOW AC.LOCATION **WITHOUT M ATTRIB
DEFINE AC.NOT.RDY.SET AS A SET RANKED BY LOW AC.LOCATION **WITHOUT M ATTRIB

NORMALLY MODE IS INTEGER

DEFINE F AS A 1-DIMENSIONAL ARRAY
DEFINE ETA AS A REAL 1-DIMENSIONAL ARRAY
DEFINE INDEX AS A 1-DIMENSIONAL ARRAY
DEFINE TYPE.AC AS A 1-DIMENSIONAL ARRAY
DEFINE SPEED.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE FUELCAP.AC AS A 1-DIMENSIONAL ARRAY
DEFINE FUELUSE.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE TTREFUEL.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE TTLQAD.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE TTRESPOT.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE TTRECOVER.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE TTARRIV.E.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE S.TTLQAD.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE S.TTRESPOT.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE S.TTRECOVER.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE S.TTARRIV.E.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE EMERGENCY.STAT.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE PRIORITY.STAT.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE TTUNLQAD.AC AS A 1-DIMENSIONAL REAL ARRAY
DEFINE NUM.LAUNCHES.AC AS A 1-DIMENSIONAL ARRAY
DEFINE SPOT AS A 1-DIMENSIONAL ARRAY
DEFINE SPOT.AC AS A 2-DIMENSIONAL ARRAY
DEFINE FLTARRAY AS A 2-DIMENSIONAL ARRAY
DEFINE HISTD.LAUNCHES AS A 2-DIMENSIONAL REAL ARRAY
DEFINE FLT.RECORD AS A 3-DIMENSIONAL REAL ARRAY

THE SYSTEM OWNS A HANGER.DECK, A BONE.FWD,
A BONE.AFT, A SET.TEMP, A DELTA.PATTERN, A PLAN,
A SCHEDULE, A REFUELER.Q, A TUG.Q, A SHIP, AN AV9.PLAN,
A LOADER.Q, A LOAD.SET, AN ELEVATOR.Q, A SPOT.Q,
AN AC.RDY.SET, AN AC.PRE.RDY.SET, AND AN AC.NOT.RDY.SET

DEFINE SCHEDULE AS A SET RANKED BY LOW FLT.TIME
DEFINE PLAN AS A SET RANKED BY LOW FLT.TIME

DEFINE AV9.PLAN AS A SET RANKED BY LOW FLT.TIME

DEFINE AC.ID, AC.TYPE, AC.LOCATION, DIST.TO.SHORE, NUM.FLTS,
TUG, AC.LAUNCH.TIME, SPOTT, AC.DESTINATION, TYPES.AC, CYCLE.NUM,
SPLIT.FLTS, AC1, AC2, AC3, AC4, AC5, AC6, AC7, AC8, F1,
AC9, AC10, AC11, AC12, FLIGHT1, FLIGHT2, FLIGHT3, SPOT1, SPOT2,
REFUELER, NUM.AV8.RDY, NUM.EMERGENCY.RECOVERIES, BONE.TOTAL,
NUM.PRIORITY.RECOVERIES, NUM.RECOVERIES, NUM.LAUNCHES,
NUM.REPLACED.AC, NUM.CANCELLED.MISSIONS,
NUM.OPEN.SPOTS, AC.SERVICE.FLAG AS INTEGER VARIABLES
DEFINE AC.FUEL.STAT, AC.LOAD.STAT, AC.PRIORITY, AC.OP.STAT, XBAR, SDEV,
DELTA.UPDATE.TIME, HANGER.EQUIV, MAX.HANGER.EQUIV, AC.FLYING.TIME,
LAST.LAUNCH.TIME, LAUNCH.DELAY, AC.DELTA.ARRIVAL.TIME, DELAY,
NUM.AV8.LOADED, TTARRIV.E, TTARRIV.H, TTARRIV.D, TTRECCVER,
TTRESPOT, TTLOAD, TTFLY, TTUNLOAD, FLT.DELAY,
LAST.RECOVERY.TIME, FLT.NUM, AC.TAKEOFF.TIME,
HELD.MISSION.LENGTH, HELD.FLYING.TIME, HELD.RECOVER.TIME,
AV8.MISSION.LENGTH, AV8.FLYING.TIME, AV8.RECOVER.TIME,
AND DELTA AS REAL VARIABLES

DEFINE FILLER AS A TEXT VARIABLE

ACCUMULATE AVG.SPOT.Q AS THE MEAN, VAR.SPOT.Q AS THE VARIANCE,
AND MAX.SPOT.Q AS THE MAXIMUM OF N.SPOT.Q
ACCUMULATE AVG.REFUELER.Q AS THE MEAN, VAR.REFUELER.Q AS THE VARIANCE,
AND MAX.REFUELER.Q AS THE MAXIMUM OF N.REFUELER.Q
ACCUMULATE AVG.REFUELER AS THE MEAN, VAR.REFUELER AS THE VARIANCE,
AND MAX.REFUELER AS THE MAXIMUM OF REFUELER
ACCUMULATE AVG.DELTA AS THE MEAN, VAR.DELTA AS THE VARIANCE,
AND MAX.DELTA AS THE MAXIMUM OF N.DELTA.PATTERN
ACCUMULATE AVG.BONE.FWD AS THE MEAN, VAR.BONE.FWD AS THE VARIANCE,
AND MAX.BONE.FWD AS THE MAXIMUM OF N.BONE.FWD
ACCUMULATE AVG.BONE.AFT AS THE MEAN, VAR.BONE.AFT AS THE VARIANCE,
AND MAX.BONE.AFT AS THE MAXIMUM OF N.BONE.AFT
ACCUMULATE AVG.BONE.TOT AS THE MEAN, VAR.BONE.TOT AS THE VARIANCE,
AND MAX.BONE.TOT AS THE MAXIMUM OF BONE.TOTAL
ACCUMULATE AVG.HANGER AS THE MEAN, VAR.HANGER AS THE VARIANCE,
AND MAX.HANGER AS THE MAXIMUM OF N.HANGER.DECK
ACCUMULATE AVG.TUG.Q AS THE MEAN, VAR.TUG.Q AS THE VARIANCE,
AND MAX.TUG.Q AS THE MAXIMUM OF N.TUG.Q
ACCUMULATE AVG.TUG AS THE MEAN, VAR.TUG AS THE VARIANCE,
AND MAX.TUG AS THE MAXIMUM OF TUG
ACCUMULATE AVG.LOADER.Q AS THE MEAN, VAR.LOADER.Q AS THE VARIANCE,
AND MAX.LOADER.Q AS THE MAXIMUM OF N.LOADER.Q
ACCUMULATE AVG.LOADER AS THE MEAN, VAR.LOADER AS THE VARIANCE,
AND MAX.LOADER AS THE MAXIMUM OF N.LOAD.SET
ACCUMULATE AVG.ELEVATOR.Q AS THE MEAN, VAR.ELEVATOR.Q AS THE VARIANCE,
AND MAX.ELEVATOR.Q AS THE MAXIMUM OF N.ELEVATOR.Q
TALLY AVG.LAUNCH.DELAY AS THE MEAN, VAR.LAUNCH.DELAY AS THE
VARIANCE, AND MAX.LAUNCH.DELAY AS THE MAXIMUM OF
LAUNCH.DELAY
TALLY AVG.HELD.FLYING.TIME AS THE MEAN, VAR.HELD.FLYING.TIME AS THE
VARIANCE, MIN.HELD.FLYING.TIME AS THE MINIMUM, AND
MAX.HELD.FLYING.TIME AS THE MAXIMUM OF HELD.FLYING.TIME
TALLY AVG.AV8.FLYING.TIME AS THE MEAN, VAR.AV8.FLYING.TIME AS THE
VARIANCE, MIN.AV8.FLYING.TIME AS THE MINIMUM, AND
MAX.AV8.FLYING.TIME AS THE MAXIMUM OF AV8.FLYING.TIME
TALLY AVG.HELD.MISSION.LENGTH AS THE MEAN, VAR.HELD.MISSION.LENGTH
AS THE VARIANCE, MIN.HELD.MISSION.LENGTH AS THE
MINIMUM, AND MAX.HELD.MISSION.LENGTH AS THE

	MAXIMUM OF HELD.MISSION.LENGTH
TALLY	AVG.AV8.MISSION.LENGTH AS THE MEAN, VAR.AV8.MISSION.LENGTH AS THE VARIANCE, MIN.AV8.MISSION.LENGTH AS THE MINIMUM, AND MAX.AV8.MISSION.LENGTH AS THE MAXIMUM OF AV8.MISSION.LENGTH
TALLY	AVG.HELO.RECOVER.TIME AS THE MEAN, VAR.HELO.RECOVER.TIME AS THE VARIANCE, MIN.HELO.RECOVER.TIME AS THE MINIMUM, AND MAX.HELO.RECOVER.TIME AS THE MAXIMUM OF HELD.RECOVER.TIME
TALLY	AVG.AV8.RECOVER.TIME AS THE MEAN, VAR.AV8.RECOVER.TIME AS THE VARIANCE, MIN.AV8.RECOVER.TIME AS THE MINIMUM, AND MAX.AV8.RECOVER.TIME AS THE MAXIMUM OF AV8.RECOVER.TIME
TALLY	AVG.TTLOAD AS THE MEAN, VAR.TTLOAD AS THE VARIANCE, MIN.TTLOAD AS THE MINIMUM, AND MAX.TTLOAD AS THE MAXIMUM OF TTLOAD
TALLY	AVG.TTRESPOT AS THE MEAN, VAR.TTRESPOT AS THE VARIANCE, MIN.TTRESPOT AS THE MINIMUM, AND MAX.TTRESPOT AS THE MAXIMUM OF TTRESPOT
TALLY	AVG.TTRECOVER AS THE MEAN, VAR.TTRECOVER AS THE VARIANCE, MIN.TTRECOVER AS THE MINIMUM, AND MAX.TTRECOVER AS THE MAXIMUM OF TTRECOVER
TALLY	AVG.TTARRIV.E AS THE MEAN, VAR.TTARRIV.E AS THE VARIANCE, MIN.TTARRIV.E AS THE MINIMUM, AND MAX.TTARRIV.E AS THE MAXIMUM OF TTARRIV.E
TALLY	AVG.TTARRIV.D AS THE MEAN, VAR.TTARRIV.D AS THE VARIANCE, MIN.TTARRIV.D AS THE MINIMUM, AND MAX.TTARRIV.D AS THE MAXIMUM OF TTARRIV.D
TALLY	AVG.TTARRIV.H AS THE MEAN, VAR.TTARRIV.H AS THE VARIANCE, MIN.TTARRIV.H AS THE MINIMUM, AND MAX.TTARRIV.H AS THE MAXIMUM OF TTARRIV.H
TALLY	AVG.TTUNLOAD AS THE MEAN, VAR.TTUNLOAD AS THE VARIANCE, MIN.TTUNLOAD AS THE MINIMUM, AND MAX.TTUNLOAD AS THE MAXIMUM OF TTUNLOAD
END	

MAIN

LET BETWEEN.V = 'TRACE'
RESERVE SPOT AS 12
USE 3 FOR INPUT

READ CYCLE.NUM
FOR I = 1 TO 9, DO
 READ FILLER,SEED.V(I)
LOOP

USE 1 FOR INPUT

OPEN 2 FOR OUTPUT
USE 2 FOR OUTPUT

PRINT 1 LINE WITH CYCLE.NUM THUS
**
FOR I = 1 TO 9, DO
 PRINT 1 LINE WITH I,SEED.V(I) THUS
SEED * = *****
LOOP

ADD 1 TO CYCLE.NUM

READ MAX.HANGER.EQUIV
PRINT 1 LINE WITH MAX.HANGER.EQUIV THUS
 HANGER IS CAPABLE OF SLASHING ** CH-46 EQUIVALENTS
READ DIST.TO.SHORE
PRINT 1 LINE WITH DIST.TO.SHORE THUS
 DISTANCE TO SHORE: ***** METERS

READ TYPES.AC
PRINT 1 LINE WITH TYPES.AC THUS
 TYPES.AC = **
RESERVE TYPE.AC AS TYPES.AC
RESERVE SPEED.AC AS TYPES.AC
RESERVE FUEL CAP.AC AS TYPES.AC
RESERVE FUELUSE.AC AS TYPES.AC
RESERVE TTREFUEL.AC AS TYPES.AC
RESERVE TTLLOAD.AC AS TYPES.AC
RESERVE S.TTLLOAD.AC AS TYPES.AC
RESERVE TTRESPOT.AC AS TYPES.AC
RESERVE S.TTRESPOT.AC AS TYPES.AC
RESERVE TTRECOVER.AC AS TYPES.AC
RESERVE S.TTRECOVER.AC AS TYPES.AC
RESERVE TTARRIV.E.AC AS TYPES.AC
RESERVE S.TTARRIV.E.AC AS TYPES.AC
RESERVE EMERGENCY.STAT.AC AS TYPES.AC
RESERVE PRIORITY.STAT.AC AS TYPES.AC
RESERVE TTUNLOAD.AC AS TYPES.AC
RESERVE SPOT.AC AS TYPES.AC BY 9
RESERVE NUM.LAUNCHES.AC AS TYPES.AC
RESERVE HISTO.LAUNCHES AS TYPES.AC BY 50
FOR I = 1 TO TYPES.AC, DO
 READ TYPE.AC(I), SPEED.AC(I), FUEL CAP.AC(I), FUELUSE.AC(I),
 TTREFUEL.AC(I), TTLLOAD.AC(I), S.TTLLOAD.AC(I), TTRECOVER.AC(I),
 S.TTRECOVER.AC(I), TTRESPOT.AC(I), S.TTRESPOT.AC(I),
 TTARRIV.E.AC(I), S.TTARRIV.E.AC(I), EMERGENCY.STAT.AC(I),
 PRIORITY.STAT.AC(I), TTUNLOAD.AC(I), SPOT.AC(I,1),

```

    SPOT.AC(I,2), SPOT.AC(I,3), SPOT.AC(I,4), SPOT.AC(I,5),
    SPOT.AC(I,6), SPOT.AC(I,7), SPOT.AC(I,8)
** PRINT 1 LINE WITH
**     TYPE.AC(I), SPEED.AC(I), FUEL CAP.AC(I), FUELUSE.AC(I),
**     TTREFUEL.AC(I), TTLOAD.AC(I), S.TTLOAD.AC(I), TTRECOVER.AC(I),
**     S.TTRECOVER.AC(I), TTRESPOT.AC(I), S.TTRESPOT.AC(I),
**     TTARRIV.E.AC(I), S.TTARRIV.E.AC(I), EMERGENCY.STAT.AC(I),
**     PRIORITY.STAT.AC(I), TTUNLOAD.AC(I), SPJT.AC(I,1),
**     SPOT.AC(I,2), SPOT.AC(I,3), SPOT.AC(I,4), SPOT.AC(I,5),
**     SPOT.AC(I,6), SPOT.AC(I,7), SPOT.AC(I,8)
**     ** * * * * *
LOOP

READ N.ACE
PRINT 1 LINE WITH N.ACE THUS
    NUM AC : **
RESERVE ETA AS N.ACE
RESERVE INDEX AS N.ACE
CREATE EVERY ACE
FOR EACH ACE, DO
    READ AC.ID(ACE), AC.LOCATION(ACE), AC.FUEL.STAT(ACE),
    AC.LOAD.STAT(ACE), AC.OP.STAT(ACE), AND AC.TYPE(ACE)
    PRINT 1 LINE WITH AC.ID(ACE), AC.LOCATION(ACE), AC.FUEL.STAT(ACE),
    AC.LOAD.STAT(ACE), AC.OP.STAT(ACE), AND AC.TYPE(ACE) THUS
    ID: ** LOC: * FUEL: **. ** LOAD: **. ** OP: **. ** TYPE: **
    LET AC.FLYING.TIME(ACE) = (AC.FUEL.STAT(ACE)/FUELUSE.AC(AC.TYPE(ACE)))
    * 60.
    IF AC.LOCATION(ACE) <= 6
        SPOT(AC.LOCATION(ACE)) = AC.ID(ACE)
    ELSE
        IF AC.LOCATION(ACE) = 7
            FILE ACE IN BONE.FWD
        ALWAYS
        IF AC.LOCATION(ACE) = 8
            FILE ACE IN BONE.AFT
        ALWAYS
        IF AC.LOCATION(ACE) = 9
            LET AC.DESTINATION(ACE) = 9
            LET AC.PRIORITY(ACE) = 1 - (AC.FUEL.STAT(ACE) * .2)
            FILE ACE IN DELTA.PATTERN
            FILE THIS ACE IN SPOT.Q
        ALWAYS
        IF AC.LOCATION(ACE) = 12
            FILE ACE IN HANGER.DECK
            IF AC.TYPE(ACE) = 2
                ADD 1.5 TO HANGER.EQUIV
            ELSE
                ADD 1 TO HANGER.EQUIV
        ALWAYS
        ALWAYS
    LET AC.LAUNCH.TIME(ACE) = 9999
    LET AC.SERVICE.FLAG(ACE) = 0
    FILE THIS ACE IN THE SHIP
LOOP
READ NUM.FLTS
LET SPLIT.FLTS = 0
RESERVE F AS (NUM.FLTS + 100)
RESERVE FLTARRAY AS (NUM.FLTS + 100) BY 6
RESERVE FLT.RECORD AS NUM.FLTS BY 10 BY 2

```

```

PRINT 1 LINE WITH NUM.FLTS THUS
* SCHEDULED FLIGHTS = **
FOR I = 1 TO NUM.FLTS, DO
  CREATE A FLIGHTE CALLED F(I)
  LET FLT.NUM(F(I)) = I
  READ FLT.TIME(F(I)), FLT.AC.TYPE(F(I)), FLT.AC.NUM(F(I))
  PRINT 1 LINE WITH I, FLT.TIME(F(I)), FLT.AC.TYPE(F(I)),
    FLT.AC.NUM(F(I))

  THUS
    I : ** FLIGHT.TIME : **** AC.TYPE : ** #AC : **
  LET FLT.RECORD(I,1,1) = FLT.TIME(F(I))
  LET FLT.RECORD(I,1,2) = FLT.AC.NUM(F(I)) + 1
  FOR J = 1 TO FLT.AC.NUM(F(I)), DO
    READ SPOTT
    FOR EACH ACE IN THE SHIP,
      WITH AC.ID(ACE) = SPOTT
      FIND THE FIRST CASE
      LET FLT.RECORD(I,J+1,1) = SPOTT
      IF AC.LAUNCH.TIME(ACE) = 9999
        LET AC.LAUNCH.TIME(ACE) = FLT.TIME(F(I))
        IF ((AC.TYPE(ACE) = 3) AND (AC.LOCATION(ACE) = 12))
          FILE ACE IN ELEVATOR.Q
          LET AC.DESTINATION(ACE) = 8
        ALWAYS
      ALWAYS
    LET FLT.ARRAY(I,J) = SPOTT
  LOOP
  FILE F(I) IN SCHEDULE
LOOP

LET NUM.OPEN.SPOTS = 0
FOR I = 1 TO 6, DO
  IF SPOT(I) = 0
    ADD I TO NUM.OPEN.SPOTS
  ALWAYS
LOOP
PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
NUM.OPEN.SPOTS = **

REMOVE FIRST FLIGHTS FROM SCHEDULE
IF (FLT.TIME(FLIGHTS) - TIME.V) > 40
  SCHEDULE A DECK.DECISION GIVING FLIGHTS IN
    (FLT.TIME(FLIGHTS) - TIME.V - 40) MINUTES
ELSE
  SCHEDULE A DECK.DECISION GIVING FLIGHTS NOW
  ALWAYS
  FILE FLIGHTS IN SCHEDULE
  READ STOP.SIM.TIME
  SCHEDULE A STOP.SIMULATION IN STOP.SIM.TIME MINUTES

  LET LAST.RECOVERY.TIME = 0
  LET LAST.LAUNCH.TIME = 0

IF N.DELTA.PATTERN > 0
  SCHEDULE A DELTA.UPDATE.CHK IN 5 MINUTES
  ALWAYS

  START SIMULATION
END

```

```

''*****
EVENT AC.LAUNCHED GIVEN AC, T.FLT.NUM
''*****

DEFINE AC, T.FLT.NUM AS INTEGER VARIABLES

PRINT 1 LINE WITH AC.ID(AC), AC.FUEL.STAT(AC), AC.LOAD.STAT(AC),
AC.OP.STAT(AC) THUS
AC ** WITH FUEL ** AND LOAD ** HAS OP.STAT **

'' SET THE AC.LAUNCHED VARIABLES AND SCHEDULE A DELTA ARRIVAL.

IF AC.LOCATION(AC) < 9
  IF T.FLT.NUM = 0
    FOR EACH FLIGHTS IN PLAN
      FOR EACH ACE IN FLT.WAVE(FLIGHTS)
        WITH ACE = AC
          FIND THE FIRST CASE
          IF FOUND
            LET T.FLT.NUM = INT.F(FLT.NUM(FLIGHTS))
          ELSE
            PRINT 1 LINE THUS
            ##ERROR## FLIGHT NOT FOUND
          ALWAYS
        ALWAYS
      ALWAYS
    IF SPOT(AC.LOCATION(AC)) = AC.ID(AC)
      LET SPOT(AC.LOCATION(AC)) = 0
      ADD 1 TO NUM.OPEN.SPOTS
    ELSE
      PRINT 1 LINE THUS
      THIS SPOT WAS CLEARED FOR AN EMERGENCY RECOVERY
      ALWAYS
      PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
    #OPEN SPOTS = ***
    LET AC.TAKEOFF.TIME(AC) = TIME.V
    IF TIME.V > LAST.LAUNCH.TIME
      LET LAST.LAUNCH.TIME = TIME.V
    ALWAYS
    ADD 1 TO NUM.LAUNCHES
    LET LAUNCH.DELAY = TIME.V - AC.LAUNCH.TIME(AC)
    PRINT 1 LINE WITH AC.ID(AC), AC.LOCATION(AC), LAUNCH.DELAY THUS
    AC ** LAUNCHES FROM ** WITH DELAY ***.
    LET AC.LOAD.STAT(AC) = 0
    LET AC.RECOVERY.TIME(AC) = 0
    LET AC.LOCATION(AC) = 10 '' AC IS IN FLIGHT
    LET AC.DESTINATION(AC) = 9

    LET TTFLY = ((DIST.TO.SHORE / SPEED.AC(AC.TYPE(AC))) / 60) * 2)
    LET TTUNLOAD = (((BETA.F(1.5, 3.0, 6)) * ((TTUNLOAD.AC(AC.TYPE(AC))) * 2.))
      + (TTUNLOAD.AC(AC.TYPE(AC)))) / 3.
    LET ETA(AC.ID(AC)) = TIME.V + TTFLY + TTUNLOAD

    IF AC.TYPE(AC) <> 3
      LET HELD.MISSION.LENGTH = TTFLY + TTUNLOAD
    ELSE ''AC IS AV8
      LET AV8.MISSION.LENGTH = TTFLY + TTUNLOAD

```

ALWAYS

```

LET AC.FUEL.STAT(AC) = (AC.FUEL.STAT(AC) - ((.15) *
(FUELUSE.AC(AC.TYPE(AC)) * (ETA(AC.ID(AC)) - TIME.V) / 60))) *
LET AC.FLYING.TIME(AC) = (AC.FUEL.STAT(AC) / FUELUSE.AC(AC.TYPE(AC))) * 60.
LET AC.PRIORITY(AC) = 1 - AC.FUEL.STAT(AC) * .2
PRINT 1 LINE WITH AC.ID(AC), ETA(AC.ID(AC)), AC.PRIORITY(AC),
AC.FLYING.TIME(AC) THUS
AC ** WILL ARRIVE TO DELTA AT ** WITH PRIORITY *** AND FLYING.TIME ***.
SCHEDULE A DELTA.ARRIVAL GIVING AC IN (ETA(AC.ID(AC)) - TIME.V) MINUTES

ADD 1 TO NUM.LAUNCHES.AC(AC.TYPE(AC))
LET HISTO.LAUNCHES(AC.TYPE(AC), NUM.LAUNCHES.AC(AC.TYPE(AC))) = TIME.V
FOR I = 1 TO FLT.RECORD(T.FLT.NUM, 1, 2)
WITH FLT.RECORD(T.FLT.NUM, I, 1) = AC.ID(AC)
FIND THE FIRST CASE
IF NONE
ADD 1 TO FLT.RECORD(T.FLT.NUM, 1, 2)
** I IS ALREADY INCREMENTED FROM THE FOR LOOP..
LET FLT.RECORD(T.FLT.NUM, I, 1) = AC.ID(AC)
ALWAYS
LET FLT.RECORD(T.FLT.NUM, I, 2) = TIME.V
PRINT 1 LINE WITH FLT.RECORD(T.FLT.NUM, 1, 1), FLT.RECORD(T.FLT.NUM, 1, 2),
FLT.RECORD(T.FLT.NUM, I, 1), FLT.RECORD(T.FLT.NUM, I, 2) THUS
FLIGHT AT *** WITH ** AC LAUNCHES AC ** AT ****.

FOR EACH FLIGHT IN THE SCHEDULE
FOR I = 1 TO FLT.AC.NUM(FLIGHT)
WITH AC.ID(AC) = FLTARRAY(FLT.NUM(FLIGHT), I)
FIND THE FIRST CASE
IF FOUND
LET AC.LAUNCH.TIME(AC) = FLT.TIME(FLIGHT)
ELSE
LET AC.LAUNCH.TIME(AC) = 9999
ALWAYS
ELSE
PRINT 1 LINE WITH AC.ID(AC) THUS
AC ** HAS ALREADY LAUNCHED
ALWAYS
RETURN
END

```

```

** *****
EVENT AC.LOADED GIVEN AC
** *****

DEFINE AC,L.TIME AS INTEGER VARIABLES

PRINT 1 LINE WITH AC.ID(AC),AC.FUEL.STAT(AC),AC.LOAD.STAT(AC),
AC.OP.STAT(AC) THUS
AC ** WITH FUEL ** AND LOAD ** HAS OP.STAT ***

LET AC.SERVICE.FLAG(AC) = 0

IF AC.LOAD.STAT(AC) < 1
LET AC.OP.STAT(AC) = AC.OP.STAT(AC) + .2*(1.0 - AC.LOAD.STAT(AC))
LET AC.LOAD.STAT(AC) = 1.0
IF AC.TYPE(AC) = 3
REMOVE AC FROM LOAD.SET
IF N.LOAD.SET < 2
IF N.LOADER.Q > 0
REMOVE FIRST ACE FROM LOADER.Q
LET XBAR = TTLOAD.AC(AC.TYPE(ACE))
LET SDEV = S.TTLOAD.AC(AC.TYPE(ACE))
LET TTLOAD = NORMAL.F(XBAR,SDEV,3)
+ (NUM.AV8S.LOADED / 4.)
ADD 1 TO NUM.AV8S.LOADED
SCHEDULE AN AC.LOADED GIVING ACE
IN TTLOAD MINUTES
LET AC.SERVICE.FLAG(ACE) = 1
FILE ACE IN LOAD.SET
PRINT 1 LINE WITH N.LOAD.SET THUS
N.LOAD.SET= **
ALWAYS
ALWAYS
ALWAYS
ELSE
PRINT 1 LINE THUS
THIS AC LOADED PREVIOUSLY
ALWAYS

IF (AC.OP.STAT(AC) < 1.01) AND (AC.OP.STAT(AC) > .99)
LET AC.OP.STAT(AC) = 1.00
ALWAYS

IF AC.OP.STAT(AC) > 1
PRINT 1 LINE THUS
**ERROR** AC.OP.STAT > 1
ALWAYS
IF AC.OP.STAT(AC) < 1
PRINT 1 LINE THUS
**ERROR** AC.OP.STAT < 1
ALWAYS
PRINT 1 LINE WITH AC.OP.STAT(AC) THUS
AC.OP.STAT(AC) : ***

PRINT 1 LINE WITH TIME.V,N.PLAN THUS
AT ***,** THERE ARE ** FLIGHTS IN THE PLAN

IF AC.LAUNCH.TIME(AC) <= (TIME.V + 40)
FOR EACH FLIGHT IN THE PLAN,

```

```

      FOR EACH ACE IN FLT.WAVE(FLIGHT)
        WITH AC.ID(AC) = AC.ID(ACE)
FIND THE FIRST CASE
IF FOUND
  PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
  AC.ID(AC) : ** AC.ID(ACE) : **
  ADD 1 TO FLT.AC.RDY(FLIGHT)
  PRINT 1 LINE WITH FLT.AC.RDY(FLIGHT) THUS
  #RDY AC IN FLIGHT = **
  IF AC.TYPE(ACE) <= 2 ** HELD READY TO LAUNCH...CHECK LAUNCH TIME
    IF FLT.AC.RDY(FLIGHT) = N.FLT.WAVE(FLIGHT)
      SCHEDULE A FLIGHT.LAUNCH GIVING FLIGHT IN .5 MINUTES
      PRINT 1 LINE THUS
      FLIGHT.LAUNCH HAS BEEN SCHEDULED
    ALWAYS
  ELSE ** AVB READY TO LAUNCH/RESPCT.....
    ** IN BOTH CASES, MUST ALSO CHECK OP.STAT, LOCATION, WAVE?
    IF (FLT.TIME(FLIGHT) - TIME.V) <= 40
      IF FLT.AC.RDY(FLIGHT) = N.FLT.WAVE(FLIGHT)
        FOR EACH ACE IN FLT.WAVE(FLIGHT), DO
          FILE ACE IN SPOT.Q
          PRINT 1 LINE WITH ACE THUS
          AC ** FILED IN SPOT.Q
        LOOP
      SCHEDULE A SPOT.OPEN NOW **THIS CAUSES THESE AVBS TO BE CONS
    ALWAYS
  ALWAYS
ELSE
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC) THUS
  **ERROR** 1:: TIME = ****.** EVENT = ** AC.ID(AC) = **
  ALWAYS
  ALWAYS
  RETURN
  END

```

```

*****
EVENT AC.REFUELED GIVEN AC
*****

DEFINE AC AS AN INTEGER VARIABLE

PRINT 1 LINE WITH AC.ID(AC),AC.FUEL.STAT(AC),AC.LOAD.STAT(AC),
AC.OP.STAT(AC) THUS
AC ** WITH FUEL ** AND LOAD ** HAS OP.STAT **

LET AC.SERVICE.FLAG(AC) = 0

IF ((AC.FUEL.STAT(AC) < 1) AND (AC.LOCATION(AC) < 9))
LET AC.OP.STAT(AC) = AC.OP.STAT(AC) + (.2)*(1.0 - AC.FUEL.STAT(AC))
LET AC.FUEL.STAT(AC) = 1.0
LET AC.FLYING.TIME(AC) = (AC.FUEL.STAT(AC) / FUELUSE.AC(AC.TYPE(AC)))
* 60.

IF AC.OP.STAT(AC) > 1
PRINT 1 LINE THUS
**ERROR** AC.OP.STAT > 1
ALWAYS
PRINT 1 LINE WITH AC.OP.STAT(AC),AC.FLYING.TIME(AC) THUS
AC.OP.STAT(AC) : ** FLYING.TIME = **.*
IF AC.LAUNCH.TIME(AC) < (TIME.V + 40)
IF AC.LOCATION(AC) < 7
LET TTLLOAD = 1. + BETA.F(1.5,3.0,3)
* TTLLOAD.AC(AC.TYPE(AC))
IF (AC.LAUNCH.TIME(AC) - TIME.V) < 10
SCHEDULE AN AC.LOADED GIVING AC
IN TTLLOAD MINUTES
ELSE
SCHEDULE AN AC.LOADED GIVING AC AT
(AC.LAUNCH.TIME(AC) - 10. + TTLLOAD)
ALWAYS
LET AC.SERVICE.FLAG(AC) = 1
ELSE
IF AC.TYPE(AC) = 1
REMOVE THIS AC FROM BONE.FWD
FILE THIS AC IN BONE.FWD
ELSE
REMOVE THIS AC FROM BONE.AFT
FILE THIS AC IN BONE.AFT
IF AC.TYPE(AC) = 3
IF M.LOAD.SET(AC) <> 1
IF N.LOAD.SET < 2
FILE AC IN LOAD.SET
PRINT 1 LINE WITH N.LOAD.SET THUS
N.LOAD.SET= **
LET XBAR = TTLLOAD.AC(AC.TYPE(AC))
LET SDEV = S.TTLLOAD.AC(AC.TYPE(AC))
LET TTLLOAD = NORMAL.F(XBAR,SDEV,3)
+ (NUM.AV8S.LOADED / 4.)
ADD 1 TO NUM.AV8S.LOADED
SCHEDULE AN AC.LOADED GIVING AC
IN TTLLOAD MINUTES
LET AC.SERVICE.FLAG(AC) = 1
ELSE
IF M.LOADER.Q(AC) <> 1
FILE AC IN LOADER.Q
ALWAYS

```

```

        ALWAYS
      ELSE
        PRINT 1 LINE THUS
        AV8 IS ALREADY BEING LOADED
        LET AC.SERVICE.FLAG(AC) = 1
      ALWAYS
    ALWAYS
  ALWAYS
  ALWAYS
  IF (AC.TYPE(AC) < 3) AND (AC.DESTINATION(AC) < 7)
    IF TUG < 4
      LET TUG = TUG + 1
      LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
      LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
      LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
      SCHEDULE AN AC.RESPOTTED GIVING AC IN TTRESPOT MINUTES
    ELSE
      FILE AC IN TUG.Q
      PRINT 1 LINE WITH AC THUS
      AC ** FILED IN TUG.Q
    ALWAYS
  ALWAYS
  ELSE
    IF AC.TYPE(AC) = 3
      IF M.LOAD.SET(AC) <> 1
        IF N.LOAD.SET < 2
          FILE AC IN LOAD.SET
          PRINT 1 LINE WITH N.LOAD.SET THUS
          N.LOAD.SET = **
          LET XBAR = TTLLOAD.AC(AC.TYPE(AC))
          LET SDEV = S.TTLLOAD.AC(AC.TYPE(AC))
          LET TTLLOAD = NORMAL.F(XBAR,SDEV,3)
            + (NUM.AV8S.LOADED / 4.)
          ADD 1 TO NUM.AV8S.LOADED
          SCHEDULE AN AC.LOADED GIVING AC
            IN TTLLOAD MINUTES
          LET AC.SERVICE.FLAG(AC) = 1
        ELSE
          IF M.LOADER.Q(AC) <> 1
            FILE AC IN LOADER.Q
          ALWAYS
        ALWAYS
      ELSE
        PRINT 1 LINE THUS
        AV8 IS ALREADY BEING LOADED
        LET AC.SERVICE.FLAG(AC) = 1
      ALWAYS
    ALWAYS
  ALWAYS
  ELSE
    PRINT 1 LINE THUS
    THIS AC HAS ALREADY BEEN REFUELED
    ALWAYS
    SUBTRACT 1 FROM REFUELER
    IF N.REFUELER.Q. > 0
      REMOVE FIRST ACE FROM REFUELER.Q
      ADD 1 TO REFUELER
      LET DELAY = NORMAL.F(1.,.25,.)
      SCHEDULE AN AC.REFUELED GIVING ACE
        IN ((1. - AC.FUEL.STAT(ACE)) *

```

```

      TTREFUEL.AC(AC.TYPE(ACE))) + DELAY MINUTES
    LET AC.SERVICE.FLAG(ACE) = 1
    IF AC.TYPE(ACE) = 3
      IF M.LOAD.SET(ACE) <> 1
        IF N.LOAD.SET < 2
          FILE ACE IN LOAD.SET
          PRINT 1 LINE WITH N.LOAD.SET THUS
        N.LOAD.SET = **
        LET XBAR = TTLOAD.AC(AC.TYPE(ACE))
        LET SDEV = S.TTLOAD.AC(AC.TYPE(ACE))
        LET TTLOAD = NORMAL.F(XBAR,SDEV,3)
          + (NUM.AV8S.LOADED / 4.)
        ADD 1 TO NUM.AV8S.LOADED
        SCHEDULE AN AC.LOADED GIVING ACE
          IN TTLOAD MINUTES
        LET AC.SERVICE.FLAG(ACE) = 1
      ELSE
        IF M.LOADER.Q(ACE) <> 1
          FILE ACE IN LOADER.Q
          ALWAYS
        ELSE
          PRINT 1 LINE THUS
    AV8 IS ALREADY BEING LOADED
      LET AC.SERVICE.FLAG(ACE) = 1
      ALWAYS
    ALWAYS
  ALWAYS
  RETURN
  END

```

```

*****
EVENT AC.RECOVERED GIVEN AC
*****

DEFINE AC, FLAG AS INTEGER VARIABLES

PRINT 1 LINE WITH AC.ID(AC),AC.FUEL.STAT(AC),AC.LOAD.STAT(AC),
      AC.OP.STAT(AC) THUS
AC ** WITH FUEL ** AND LOAD ** HAS OP.STAT **

REMOVE THIS AC FROM DELTA.PATTERN
ADD 1 TO NUM.RECOVERIES
LET AC.RECOVERY.TIME(AC) = TIME.V
IF AC.TAKEOFF.TIME(AC) > 0
  IF AC.TYPE(AC) <> 3
    LET HELD.FLYING.TIME = AC.RECOVERY.TIME(AC) - AC.TAKEOFF.TIME(AC)
    LET HELD.RECOVER.TIME = AC.RECOVERY.TIME(AC)
      - AC.DELTA.ARRIVAL.TIME(AC)
  ELSE
    LET AV8.FLYING.TIME = AC.RECOVERY.TIME(AC) - AC.TAKEOFF.TIME(AC)
    LET AV8.RECOVER.TIME = AC.RECOVERY.TIME(AC)
      - AC.DELTA.ARRIVAL.TIME(AC)
  ALWAYS
  ALWAYS
  LET AC.LOCATION(AC) = AC.DESTINATION(AC)
  LET SPOT.AC(AC.TYPE(AC),I) = AC.ID(AC)
  LET AC.PRIORITY(AC) = 0
  LET AC.OP.STAT(AC) = AC.OP.STAT(AC) - (.2)*(1. - AC.LOAD.STAT(AC))
      - (.2)*(1. - AC.FUEL.STAT(AC))

  LET FLAG = 0
  FOR I = 1 TO SPOT.AC(AC.TYPE(AC),7),
    WITH SPOT.AC(AC.TYPE(AC),I) = AC.LOCATION(AC)
  FIND THE FIRST CASE
  IF NONE
    LET FLAG = 1
  ALWAYS

  IF N.AV8.PLAN > 0
    REMOVE THE FIRST FLIGHTS FROM AV8.PLAN
    LET AV8.LAUNCH.TIME = FLT.TIME(FLIGHTS)
    FILE THIS FLIGHTS IN AV8.PLAN
  ELSE
    LET AV8.LAUNCH.TIME = 9999
  ALWAYS

  PRINT 1 LINE WITH AC.LAUNCH.TIME(AC), AC.LOCATION(AC), N.SPOT.Q,
        NUM.OPEN.SPOTS, AV8.LAUNCH.TIME THUS
  TO TIME= **** LOC= ** N.SPOT.Q= ** NUM.OPEN.SPOTS= ** AV8 TO TIME= **

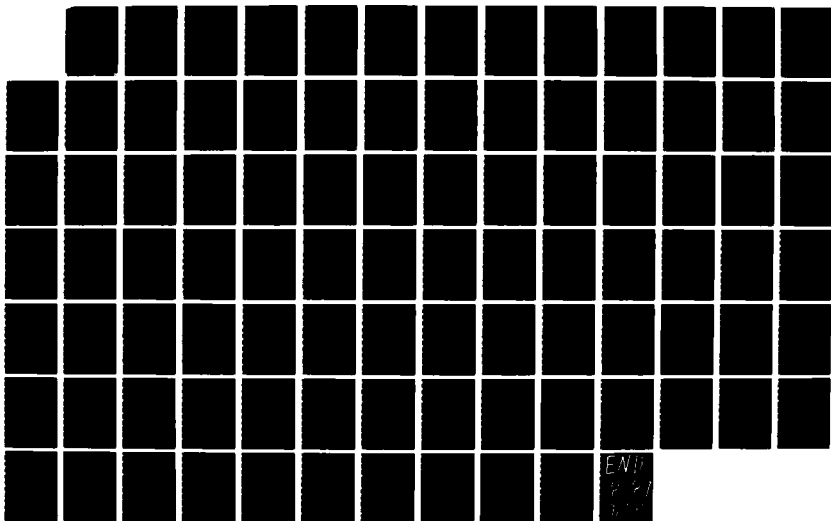
  IF (AC.LAUNCH.TIME(AC) > (40+TIME.V))
    OR ((AV8.LAUNCH.TIME < AC.LAUNCH.TIME(AC))
      AND (AC.LOCATION(AC) <= 2)) OR (FLAG = 1)
  IF AC.TYPE(AC) = 1
    IF N.BONE.FWD < 5
      LET AC.DESTINATION(AC) = 7
      IF TUG < 4
        LET TUG = TUG + 1
        LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))

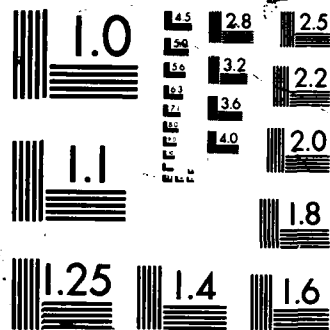
```

AD-A182 192

SIMULATION AND ANALYSIS OF FLIGHT DECK OPERATIONS ON AN 2/2
LHA(U) CENTER FOR NAVAL ANALYSES ALEXANDRIA VA MARINE
CORPS OPERATIONS ANALYSIS GROUP S M GATES JUN 87
CNA-PP-456 N00014-87-C-0001 F/G 1/2 NL

UNCLASSIFIED





PHOTOCOPY RESOLUTION TEST CHART

```

        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE A BONE.ARRIVAL GIVING AC IN TTRESPOT MINUTES
    ELSE
        FILE THIS AC IN TUG.Q
    PRINT 1 LINE WITH AC THUS
    AC ** FILED IN TUG.Q
        ALWAYS
        FILE AC IN BONE.FWD
    PRINT 1 LINE WITH AC THUS
    AC ** FILED IN BONE
        ELSE IF N.BONE.FWD >= 5
        IF AC.LAUNCH.TIME(AC) < 9999
            LET AC.DESTINATION(AC) = 7
            IF TUG < 4
                LET TUG = TUG + 1
                LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
                LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
                LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
                SCHEDULE A BONE.ARRIVAL GIVING AC IN TTRESPOT MINUTES
            ELSE
                FILE THIS AC IN TUG.Q
    PRINT 1 LINE WITH AC THUS
    AC ** FILED IN TUG.Q
                ALWAYS
                FILE AC IN BONE.FWD
    PRINT 1 LINE WITH AC THUS
    AC ** FILED IN BONE
                REMOVE THE LAST ACE FROM BONE.FWD
                IF (AC.LAUNCH.TIME(ACE)=9999)
                    LET AC.DESTINATION(ACE) = 12
                    IF (SPOT(11) = 0) AND (HANGER.EQUIV < MAX.HANGER.EQUIV)
                        LET SPOT(11) = -AC.ID(ACE)
                        IF TUG < 4
                            ADD 1 TO TUG
                            LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
                            LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
                            LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                            SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
                                IN TTARRIV.E MINUTES
                        ELSE
                            FILE ACE IN TUG.Q
    PRINT 1 LINE WITH ACE THUS
    AC ** FILED IN TUG.Q
                            ALWAYS
                            ELSE
                                IF M.ELEVATOR.Q(ACE) <> 1
                                    FILE ACE IN ELEVATOR.Q
                                ALWAYS
    PRINT 1 LINE WITH ACE THUS
    AC ** FILED IN ELEVATOR.Q
                            ALWAYS
                            FILE ACE IN BONE.FWD
    PRINT 1 LINE WITH ACE THUS
    AC ** FILED IN BONE
                            ELSE
                                PRINT 1 LINE WITH AC.ID(AC),AC.LOCATION(AC) THUS
                                AC ** AT LOCATION ** CANNOT RESPOT TO FWD.BONE...5 AC ALREADY THERE
                                LET AC.DESTINATION(AC) = 12
                                IF (SPOT(11) = 0) AND (HANGER.EQUIV < MAX.HANGER.EQUIV)

```

```

    LET SPOT(11) = -AC.ID(AC)
    IF TUG < 4
        ADD 1 TO TUG
        LET XBAR = TTARRIV.E.AC(AC.TYPE(AC))
        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(AC))
        LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
        SCHEDULE AN ELEVATOR.ARRIVAL GIVING AC
            IN TTARRIV.E MINUTES
    ELSE
        FILE AC IN TUG.Q
PRINT 1 LINE WITH AC THUS
AC ** FILED IN TUG.Q
    ALWAYS
    ELSE
        IF M.ELEVATOR.Q(AC) <> 1
            FILE THIS AC IN ELEVATOR.Q
        ALWAYS
PRINT 1 LINE WITH AC THUS
AC ** FILED IN ELEVATOR.Q
    ALWAYS
    ALWAYS
    ALWAYS
    ELSE
        IF N.BONE.AFT < 7
            LET AC.DESTINATION(AC) = 8
            IF AC.TYPE(AC) < 3
                IF TUG < 4
                    LET TUG = TUG + 1
                    LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
                    LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
                    LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
                    SCHEDULE A BONE.ARRIVAL GIVING AC
                        IN TTRESPOT MINUTES
                ELSE
                    FILE THIS AC IN TUG.Q
PRINT 1 LINE WITH AC THUS
AC ** FILED IN TUG.Q
            ALWAYS
            ELSE
                LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
                LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
                LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
                SCHEDULE A BONE.ARRIVAL GIVING AC IN TTRESPOT/2 MINUTES
            ALWAYS
            FILE AC IN BONE.AFT
PRINT 1 LINE WITH AC THUS
AC ** FILED IN BONE
        ELSE IF N.BONE.AFT >= 7
            IF AC.LAUNCH.TIME(AC) < 9999
                LET AC.DESTINATION(AC) = 8
                LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
                LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
                LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
                IF AC.TYPE(AC) < 3
                    IF TUG < 4
                        LET TUG = TUG + 1
                        SCHEDULE A BONE.ARRIVAL GIVING AC
                            IN TTRESPOT MINUTES
                    ELSE
                        FILE THIS AC IN TUG.Q

```

```

PRINT 1 LINE WITH AC THUS
AC ** FILED IN TUG.Q
    ALWAYS
    ELSE ** AC IS AVB
        SCHEDULE A BONE.ARRIVAL GIVING AC
            IN TTRESPOT MINUTES
    ALWAYS
    FILE AC IN BONE.AFT
PRINT 1 LINE WITH AC THUS
AC ** FILED IN BONE
    REMOVE THE LAST ACE FROM BONE.AFT
    IF (AC.LAUNCH.TIME(ACE)=9999)
        LET AC.DESTINATION(ACE) = 12
        IF (SPOT(11) = 0) AND (HANGER.EQUIV < MAX.HANGER.EQUIV)
            LET SPOT(11) = -AC.ID(ACE)
            IF TUG < 4
                ADD 1 TO TUG
                LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
                LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
                LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
                    IN TTARRIV.E MINUTES
            ELSE
                FILE ACE IN TUG.Q
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q
    ALWAYS
    ELSE
        IF M.ELEVATOR.Q(ACE) <> 1
            FILE ACE IN ELEVATOR.Q
    ALWAYS
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q
    ALWAYS
    ALWAYS
    FILE ACE IN BONE.AFT
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN BONE
    ELSE
        PRINT 1 LINE WITH AC.ID(AC),AC.LOCATION(AC) THUS
        AC ** AT LOCATION ** CANNOT RESPOT TO AFT.BONE...7 AC ALREADY THERE
        LET AC.DESTINATION(AC) = 12
        IF (SPOT(11) = 0) AND (HANGER.EQUIV < MAX.HANGER.EQUIV)
            LET SPOT(11) = -AC.ID(AC)
            IF TUG < 4
                ADD 1 TO TUG
                LET XBAR = TTARRIV.E.AC(AC.TYPE(AC))
                LET SDEV = S.TTARRIV.E.AC(AC.TYPE(AC))
                LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                SCHEDULE AN ELEVATOR.ARRIVAL GIVING AC
                    IN TTARRIV.E MINUTES
            ELSE
                FILE AC IN TUG.Q
PRINT 1 LINE WITH AC THUS
AC ** FILED IN TUG.Q
    ALWAYS
    ELSE
        IF M.ELEVATOR.Q(AC) <> 1
            FILE AC IN ELEVATOR.Q
    ALWAYS

```

```

PRINT 1 LINE WITH AC THUS
AC ** FILED IN ELEVATOR.Q
    ALWAYS
    ALWAYS
    ALWAYS
    PRINT 1 LINE WITH AC.ID(AC),AC.LOCATION(AC),AC.DESTINATION(AC),
        TUG,N.TUG.Q,N.ELEVATOR.Q THUS
AC ** IS AT ** WITH DEST ** TUG= ** N.TUG.Q= ** N.ELEVATOR.Q= **
ELSE
    PRINT 1 LINE WITH REFUELER,N.REFUELER.Q THUS
    REFUELER= ** N.REFUELER.Q = **
    IF AC.TYPE(AC) < 3
        LET AC.DESTINATION(AC) = 10
        IF REFUELER < 4
            ADD 1 TO REFUELER
            LET DELAY = NORMAL.F(1,.,.25,4)
            SCHEDULE AN AC.REFUELED GIVING AC **FOR HELOS ONLY*****
            IN ((1. - AC.FUEL.STAT(AC)) *
            TTREFUEL.AC(AC.TYPE(AC))) + DELAY MINUTES
            LET AC.SERVICE.FLAG(AC) = 1
        ELSE
            FILE THIS AC IN REFUELER.Q
            ALWAYS
        ELSE
            LET AC.DESTINATION(AC) = 8
            LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
            LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
            LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
            SCHEDULE A BONE.ARRIVAL GIVING AC IN TTRESPOT/2 MINUTES **AV8S
            FILE AC IN BONE.AFT
    PRINT 1 LINE WITH AC THUS
    AC ** FILED IN BONE
    ALWAYS
    ALWAYS
    RETURN
END

```

```

** *****
EVENT AC.RESPOTTED GIVEN AC
** *****

```

```

DEFINE AC, OPEN.SPOT, AND COUNTER AS INTEGER VARIABLES
PRINT 1 LINE WITH AC.ID(AC), AC.FUEL.STAT(AC), AC.LOAD.STAT(AC),
AC.OP.STAT(AC) THUS
AC ** WITH FUEL **. ** AND LOAD **. ** HAS OP.STAT **. **

```

```

** A TUG BECOMES AVAILABLE.

```

```

IF AC.TYPE(AC) < 3
  LET TUG = TUG - 1
  IF AC.LOCATION(AC) < 7
    LET SPOT(AC.LOCATION(AC)) = 0
    SCHEDULE A SPOT.OPEN NOW.
  ALWAYS
ALWAYS

```

```

** REMOVE THIS RESPOTTED AIRCRAFT FROM THE BONE IF IT WAS RESPOTTED FROM
** THAT AREA.

```

```

IF AC.LOCATION(AC) = 7
  REMOVE THIS AC FROM THE BONE.FWD
  ALWAYS
IF AC.LOCATION(AC) = 8
  REMOVE THIS AC FROM THE BONE.AFT
  ALWAYS

```

```

LET AC.LOCATION(AC) = AC.DESTINATION(AC)
LET SPOT(AC.LOCATION(AC)) = AC.ID(AC)
LET AC.DESTINATION(AC) = 10
IF AC.TYPE(AC) < 3
  IF AC.FUEL.STAT(AC) = 1
    LET TTLOAD = 1. + BETA.F(1.5, 3.0, 3)
    * TTLOAD.AC(AC.TYPE(AC))
    IF AC.LAUNCH.TIME(AC) - TIME.V < 10
      SCHEDULE AN AC.LOADED GIVING AC
      IN TTLOAD MINUTES
    ELSE
      SCHEDULE AN AC.LOADED GIVING AC AT
      ((AC.LAUNCH.TIME(AC) - 10. + TTLOAD))
    ALWAYS
    LET AC.SERVICE.FLAG(AC) = 1
  ELSE
    IF REFUELER < 4
      ADD 1 TO REFUELER
      LET DELAY = NORMAL.F(1.0, 25, 4)
      SCHEDULE AN AC.REFUELED GIVING AC IN
      ((1. - AC.FUEL.STAT(AC)) *
      TTREFUEL.AC(AC.TYPE(AC))) + DELAY MINUTES
      LET AC.SERVICE.FLAG(AC) = 1
    ELSE
      FILE THIS AC IN REFUELER.Q
    ALWAYS
  ALWAYS

```

```

** DETERMINE IF THE NOW AVAILABLE TUG CAN BE USED BY WAITING AIRCRAFT IN
** THE TUG QUEUE.

```

```

IF N.TUG.Q > 0 AND TUG < 4
  REMOVE FIRST ACE FROM TUG.Q
  PRINT 1 LINE WITH ACE THUS
AC ** REMOVED FROM TUG.Q
  LET TUG = TUG + 1
  IF AC.DESTINATION(ACE) = 7 OR AC.DESTINATION(ACE) = 8
    LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
    LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
    LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
    SCHEDULE A BONE.ARRIVAL GIVING ACE IN TTRESPOT MINUTES
  ELSE
    IF AC.DESTINATION(ACE) < 7
      LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
      LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
      LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
      SCHEDULE AN AC.RESPOTTED GIVING ACE IN TTRESPOT MINUTES
    ELSE
      LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
      LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
      LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
      SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
        IN TTARRIV.E MINUTES
    ALWAYS
  ALWAYS
  IF (AC.LOCATION(ACE) = 11)
    LET SPOT(11) = 0
  ALWAYS
  ALWAYS
  IF ((SPOT(11) = 0) AND (N.ELEVATOR.Q > 0))
    REMOVE THE FIRST ACE FROM ELEVATOR.Q
    IF AC.DESTINATION(ACE) > 0
      IF AC.DESTINATION(ACE) = 12
        IF (HANGER.EQUIV + 1) < (MAX.HANGER.EQUIV)
          LET SPOT(11) = -AC.ID(ACE)
          IF TUG < 4
            ADD 1 TO TUG
            LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
            LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
            LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
            SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
              IN TTARRIV.E MINUTES
          ELSE
            FILE ACE IN TUG.Q
        PRINT 1 LINE WITH ACE THUS
        AC ** FILED IN TUG.Q
      ALWAYS
    ELSE
      IF N.ELEVATOR.Q(ACE) <> 1
        FILE ACE IN ELEVATOR.Q
      ALWAYS
    PRINT 1 LINE WITH ACE THUS
    AC ** FILED IN ELEVATOR.Q
    FOR EACH ACE IN ELEVATOR.Q
      WITH AC.DESTINATION(ACE) < 12
      FIND THE FIRST CASE
      IF FOUND
        REMOVE THIS ACE FROM ELEVATOR.Q
        LET SPOT(11) = -AC.ID(ACE)

```

```

        LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
        LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
        SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
        IN TTARRIV.E MINUTES
    ALWAYS
    ALWAYS
ELSE
    LET SPOT(11) = -AC.ID(ACE)
    LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
    LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
    LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
    SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
    IN TTARRIV.E MINUTES
    ALWAYS
ELSE
    IF AC.TYPE(ACE) = 1
        IF N.BONE.FWD < 5
            FILE ACE IN BONE.FWD
            LET AC.DESTINATION(ACE) = 7
            LET SPOT(11) = -AC.ID(ACE)
            LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
            LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
            LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
            SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
            IN TTARRIV.E MINUTES
        ALWAYS
    ELSE
        IF N.BONE.AFT < 7
            FILE ACE IN BONE.AFT
            LET AC.DESTINATION(ACE) = 8
            LET SPOT(11) = -AC.ID(ACE)
            LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
            LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
            LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
            SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
            IN TTARRIV.E MINUTES
        ALWAYS
    ALWAYS
    ALWAYS
    ALWAYS
ELSE **CHECK IF AV8 FLIGHT RDY TO LAUNCH...SCHED LAUNCH IF RDY
    IF AC.LOCATION(AC) <= 2
        PRINT 1 LINE WITH AC.ID(AC),TIME.V,AC.LOCATION(AC) THUS
        AV8 ** RDY TO LAUNCH AT **** FROM SPOT **
        FOR EACH FLIGHT IN AV8.PLAN
            WITH FLT.TIME(FLIGHT) = AC.LAUNCH.TIME(AC)
            FIND THE FIRST CASE
            IF FOUND
                ADD 1 TO NUM.AV8.RDY
                PRINT 1 LINE WITH FLT.TIME(FLIGHT),NUM.AV8.RDY THUS
        AV8 FLIGHT SCHED TO LAUNCH AT **** HAS ** AV8'S RDY ON SPOTS
        IF NUM.AV8.RDY = 2
            LET NUM.AV8.RDY = 0
            SCHEDULE A FLIGHT.LAUNCH GIVING FLIGHT IN
            UNIFORM.F(.1,.5,9) MINUTES
            PRINT 1 LINE WITH FLT.TIME(FLIGHT) THUS
        AV8 FLIGHT SCHE TO LAUNCH AT **** HAS BEEN SENT TO FLIGHT.LAUNCH
        FOR EACH ACE IN FLT.WAVE(FLIGHT), DO

```

```

        IF N.SPOT.Q(ACE) = 1
            REMOVE THIS ACE FROM SPOT.Q
        ALWAYS
    LOOP
    ALWAYS
ELSE
    PRINT 1 LINE WITH AC.ID(AC) THUS
    **ERROR IN AC.RESPOTTED WITH AC **
    ALWAYS
ELSE **NEED SOME OTHER ACTION FOR RESPOTTED AV8
    PRINT 1 LINE WITH AC.ID(AC),AC.LOCATION(AC) THUS
    ** ERROR AC ** (AV8) RESPOTTED TO ** ...NEED SOME ACTION**
    ALWAYS
ALWAYS

```

```

RETURN
END

```

```

*****
EVENT BONE.ARRIVAL GIVEN AC
*****

DEFINE AC AS INTEGER VARIABLES

PRINT 1 LINE WITH AC.ID(AC),AC.FUEL.STAT(AC),AC.LOAD.STAT(AC),
AC.OP.STAT(AC) THUS
AC ** WITH FUEL **. ** AND LOAD **. ** HAS OP.STAT **. **

IF (AC.TYPE(AC) < 3) OR (AC.LOCATION(AC) = 11)
  LET TUG = TUG - 1
  ALWAYS

IF AC.LOCATION(AC) < 7
  IF SPOT(AC.LOCATION(AC)) = AC.ID(AC)
    LET SPOT(AC.LOCATION(AC)) = 0
    ADD 1 TO NUM.OPEN.SPOTS
    SCHEDULE A SPOT.OPEN NOW
  ELSE **AC MOVED TO BONE FOR EMERGENCY RECOVERY
    ALWAYS
  ELSE **AC IS COMING FROM THE HANGER DECK
    ALWAYS

IF AC.LAUNCH.TIME(AC) = 0
  LET AC.LAUNCH.TIME(AC) = 9999
ELSE
  IF (AC.LAUNCH.TIME(AC) <= (TIME.V + 40)) AND ((AC.TYPE(AC) < 3)
    OR (AC.OP.STAT(AC) = 1.0))
    AND (M.SPOT.Q(AC) <> 1)

    FILE AC IN SPOT.Q
    ALWAYS
  ALWAYS
  IF AC.TYPE(AC) = 1
    LET AC.LOCATION(AC) = 7
    IF M.BONE.FWD(AC) <> 1
      FILE THIS AC IN BONE.FWD
    ALWAYS
  ELSE
    LET AC.LOCATION(AC) = 8
    IF M.BONE.AFT(AC) <> 1
      FILE THIS AC IN BONE.AFT
    ALWAYS
  ALWAYS

LET BONE.TOTAL = M.BONE.AFT + M.BONE.FWD

PRINT 1 LINE WITH M.BONE.FWD, M.BONE.AFT, BONE.TOTAL,
NUM.OPEN.SPOTS THUS
M.BONE.FWD= ** M.BONE.AFT= ** BONE.TOTAL= ** NUM.OPEN.SPOTS= **

IF AC.OP.STAT(AC) <> 0 AND AC.FUEL.STAT(AC) < 1.0
  IF REFUELER < 4
    ADD 1 TO REFUELER
    LET DELAY = NORMAL.F(1.,.25,4)
    SCHEDULE AN AC.REFUELED GIVING AC
    IN ((1. - AC.FUEL.STAT(AC)) *
      TTREFUEL.AC(AC.TYPE(AC))) + DELAY MINUTES
    LET AC.SERVICE.FLAG(AC) = 1

```

```

IF AC.TYPE(AC) = 3  'AV8S CAN LOAD WHILE REFUELING
  IF M.LOAD.SET(AC) <> 1
    IF N.LOAD.SET < 2
      FILE AC IN LOAD.SET
      PRINT 1 LINE WITH N.LOAD.SET THUS
N.LOAD.SET= **
      LET XBAR = TTLOAD.AC(AC.TYPE(AC))
      LET SDEV = S.TTLOAD.AC(AC.TYPE(AC))
      LET TTLOAD = NORMAL.F(XBAR,SDEV,3)
        + (NUM.AV8S.LOADED / 4.)
      ADD 1 TO NUM.AV8S.LOADED
      SCHEDULE AN AC.LOADED GIVING AC
        IN TTLOAD MINUTES
      LET AC.SERVICE.FLAG(AC) = 1
    ELSE
      IF M.LOADER.Q(AC) <> 1
        FILE AC IN LOADER.Q
        ALWAYS
      ALWAYS
    ELSE
      PRINT 1 LINE THUS
AV8 IS ALREADY BEING LOADED
      LET AC.SERVICE.FLAG(AC) = 1
      ALWAYS
    ALWAYS
  ELSE
    FILE AC IN REFUELER.Q
    ALWAYS
  ALWAYS

  IF (N.TUG.Q > 0) AND (TUG < 4)
    REMOVE FIRST ACE FROM TUG.Q
    PRINT 1 LINE WITH ACE THUS
AC ** REMOVED FROM TUG.Q
    LET TUG = TUG + 1
    IF AC.DESTINATION(ACE) = 7 OR AC.DESTINATION(ACE) = 8
      LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
      LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
      LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
      SCHEDULE A BONE.ARRIVAL GIVING ACE IN TTRESPOT MINUTES
    ELSE
      IF AC.DESTINATION(ACE) < 7
        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE AN AC.RESPOTTED GIVING ACE IN TTRESPOT MINUTES
      ELSE ' AC.DESTINATION(ACE) = 12
        LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
        LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
        SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
          IN TTARRIV.E MINUTES
        ALWAYS
      ALWAYS
    IF (AC.LOCATION(ACE)=11) AND (N.ELEVATOR.Q > 0)
      REMOVE THE FIRST ACE FROM ELEVATOR.Q
      IF AC.DESTINATION(ACE) = 12
        IF (HANGER.EQUIV + 1) < (MAX.HANGER.EQUIV)
          LET SPOT(11) = -AC.ID(ACE)
          IF TUG < 4

```

```

        ADD 1 TO TUG
        LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
        LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
        SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
            IN TTARRIV.E MINUTES
    ELSE
        FILE ACE IN TUG.Q
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q
    ALWAYS
    ELSE
        IF M.ELEVATOR.Q(ACE) <> 1
            FILE ACE IN ELEVATOR.Q
        ALWAYS
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q
        FOR EACH ACE IN ELEVATOR.Q
            WITH AC.DESTINATION(ACE) < 12
            FIND THE FIRST CASE
            IF FOUND
                REMOVE THIS ACE FROM ELEVATOR.Q
                LET SPOT(11) = -AC.ID(ACE)
                LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
                LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
                LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
                    IN TTARRIV.E MINUTES
            ALWAYS
        ALWAYS
    ELSE
        LET SPOT(11) = -AC.ID(ACE)
        LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
        LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
        SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
            IN TTARRIV.E MINUTES
        ALWAYS
    ALWAYS
    ALWAYS
    IF N.BONE.FWD > 7
        REMOVE LAST ACE FROM BONE.FWD
        IF (AC.LAUNCH.TIME(ACE)=9999)
            LET AC.DESTINATION(ACE) = 12
            IF (SPOT(11) = 0) AND (HANGER.EQUIV < MAX.HANGER.EQUIV)
                LET SPOT(11) = -AC.ID(ACE)
            IF TUG < 4
                ADD 1 TO TUG
                LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
                LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
                LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
                    IN TTARRIV.E MINUTES
            ELSE
                FILE ACE IN TUG.Q
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q
        ALWAYS
    ELSE
        IF M.ELEVATOR.Q(ACE) <> 1

```

```

                FILE ACE IN ELEVATOR.Q
            ALWAYS
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q
        ALWAYS
        ALWAYS
        FILE ACE IN BONE.FWD
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN BONE
        ALWAYS
        IF M.BONE.AFT > 9
            REMOVE LAST ACE FROM BONE.AFT
            IF (AC.LAUNCH.TIME(ACE)=9999)
                LET AC.DESTINATION(ACE) = 12
                IF (SPOT(11) = 0) AND (CHANGER.EQUIV < MAX.HANGER.EQUIV)
                    LET SPOT(11) = -AC.ID(ACE)
                    IF TUG < 4
                        ADD 1 TO TUG
                        LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
                        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
                        LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                        SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
                            IN TTARRIV.E MINUTES
                    ELSE
                        FILE ACE IN TUG.Q
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q
                ALWAYS
                ELSE
                    IF M.ELEVATOR.Q(ACE) <> 1
                        FILE ACE IN ELEVATOR.Q
                ALWAYS
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q
                ALWAYS
                ALWAYS
                FILE ACE IN BONE.AFT
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN BONE
                ALWAYS
                RETURN
            END

```

```

*****
EVENT DECK.ARRIVAL GIVEN AC
*****

```

```

DEFINE AC AS AN INTEGER VARIABLE

```

```

LET AC.LOCATION(AC) = 11
LET SPOT(11) = AC.ID(AC)

```

```

IF AC.TYPE(AC) = 2
  SUBTRACT 1.5 TO HANGER.EQUIV
ELSE
  SUBTRACT 1.0 TO HANGER.EQUIV
ALWAYS

```

```

PRINT 1 LINE WITH HANGER.EQUIV THUS
HANGER.EQUIV = **.*

```

```

IF TUG < 4
  ADD 1 TO TUG
  IF AC.DESTINATION(AC) < 7
    LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
    LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
    LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
    SCHEDULE AN AC.RESPOTTED GIVING AC IN TTRESPOT MINUTES
  ELSE
    LET XBAR = TTRESPOT.AC(AC.TYPE(AC))
    LET SDEV = S.TTRESPOT.AC(AC.TYPE(AC))
    LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
    SCHEDULE A BONE.ARRIVAL GIVING AC IN TTRESPOT MINUTES
  ALWAYS
  LET SPOT(11) = 0
  IF M.ELEVATOR.Q > 0
    REMOVE THE FIRST ACE FROM ELEVATOR.Q
    IF AC.DESTINATION(ACE) = 12
      IF (HANGER.EQUIV + 1) < (MAX.HANGER.EQUIV)
        LET SPOT(11) = -AC.ID(ACE)
        IF TUG < 4
          ADD 1 TO TUG
          LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
          LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
          LET TTARRIV.E = NORMAL.F(XBAR,SDEV,3)
          SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
            IN TTARRIV.E MINUTES
        ELSE
          FILE ACE IN TUG.Q
      PRINT 1 LINE WITH ACE THUS
      AC ** FILED IN TUG.Q
      ALWAYS
    ELSE
      IF M.ELEVATOR.Q(ACE) <> 1
        FILE ACE IN ELEVATOR.Q
      ALWAYS
  PRINT 1 LINE WITH ACE THUS
  AC ** FILED IN ELEVATOR.Q
  FOR EACH ACE IN ELEVATOR.Q
    WITH AC.DESTINATION(ACE) < 12

```

```

FIND THE FIRST CASE
IF FOUND
  REMOVE THIS ACE FROM ELEVATOR.Q
  LET SPOT(11) = -AC.ID(ACE)
  LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
  LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
  LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
  SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
    IN TTARRIV.E MINUTES
  ALWAYS
  ALWAYS
ELSE
  LET SPOT(11) = -AC.ID(ACE)
  LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
  LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
  LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
  SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
    IN TTARRIV.E MINUTES
  ALWAYS
  ALWAYS
ELSE
  FILE AC IN TUG.Q
  PRINT 1 LINE WITH AC THUS
  AC ** FILED IN TUG.Q
  ALWAYS
  RETURN
  END

```

```

** *****
EVENT DECK.DECISION GIVEN FLIGHT
** *****

```

```

DEFINE COUNTER,FLIGHT,AC,T.AC,SPOTT,LOSPOT,AV8.LAUNCH.TIME,
      TT.NUM.AC AS INTEGER VARIABLES
DEFINE INTERVAL AS A REAL VARIABLE

```

```

REMOVE THIS FLIGHT FROM SCHEDULE

```

```

IF N.AV8.PLAN > 0
  REMOVE THE FIRST FLIGHT FROM AV8.PLAN
  LET AV8.LAUNCH.TIME = FLT.TIME(FLIGHT)
  FILE THIS FLIGHT IN AV8.PLAN
ELSE
  LET AV8.LAUNCH.TIME = 9999
  ALWAYS

```

```

** CHECK THAT DESIGNATED AC ARE STILL AVAILABLE FOR THIS FLIGHT

```

```

LET COUNTER = 0
PRINT 1 LINE WITH FLT.AC.NUM(FLIGHT),FLT.AC.TYPE(FLIGHT) THUS
THERE ARE * AC OF TYPE * IN THIS FLIGHT

```

```

FOR J = 1 TO FLT.AC.NUM(FLIGHT), DO
  FOR EACH ACE IN THE SHIP,
    WITH AC.ID(ACE) = FLTARRAY(FLT.NUM(FLIGHT),J)
    FIND THE FIRST CASE
    IF FOUND

```

```

  PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),AC.LAUNCH.TIME(ACE),
    ETA(AC.ID(ACE)) THUS

```

```

AC ** . AT ** SCHED TO LAUNCH AT **** AND RETURN AT ****
IF ((ETA(AC.ID(ACE)) + 5 <= FLT.TIME(FLIGHT)) OR
  (AC.LOCATION(ACE) < 9)) AND (AC.LAUNCH.TIME(ACE) =
  FLT.TIME(FLIGHT))
  ADD 1 TO COUNTER
  FILE THIS ACE IN FLT.WAVE(FLIGHT)
  LET FLTARRAY(FLT.NUM(FLIGHT),J) = 0

```

```

PRINT 1 LINE WITH AC.ID(ACE) AND COUNTER THUS
AC : ** COUNTER : **

```

```

IF ((AC.LOCATION(ACE) < 7) AND (AC.DESTINATION(ACE) = 10)) ** OTHERWISE
  IF AC.SERVICE.FLAG(ACE) = 0
    IF AC.FUEL.STAT(ACE) < 1.0
      IF REFUELER < 4
        ADD 1 TO REFUELER
        PRINT 1 LINE WITH REFUELER THUS
        REFUELER = **
        LET DELAY = NORMAL.F(1...25,4)
        SCHEDULE AN AC.REFUELED GIVING ACE
          IN ((1. - AC.FUEL.STAT(ACE)) *
            TTREFUEL.AC(AC.TYPE(ACE))) +
            DELAY MINUTES
        LET AC.SERVICE.FLAG(ACE) = 1
      ELSE
        FILE ACE IN REFUELER.Q
      ALWAYS
    ELSE

```

```

    LET TTLQAD = 1. + BETA.F(1.5,3.0,3)
      * TTLQAD.AC(AC.TYPE(ACE))
    IF (AC.LAUNCH.TIME(ACE) - TIME.V) < 10
      SCHEDULE AN AC.LOADED GIVING ACE
        IN TTLQAD MINUTES
    ELSE
      SCHEDULE AN AC.LOADED GIVING ACE
        AT AC.LAUNCH.TIME(ACE)
        - 10. + TTLQAD
    ALWAYS
    LET AC.SERVICE.FLAG(ACE) = 1
  ALWAYS
ALWAYS
ALWAYS

```

```

..      IF AC.LOCATION(ACE) = 9
..      IF AC.RECOVERY.TIME(ACE) = 0
..      REMOVE THIS ACE FROM THE DELTA.PATTERN
..      FILE THIS ACE IN DELTA.PATTERN
..      IF M.SPOT.Q(ACE) = 1
..      REMOVE THIS ACE FROM SPOT.Q
..      FILE THIS ACE IN SPOT.Q
..      ALWAYS
..      ALWAYS
..      ALWAYS

```

```

    IF (AC.LOCATION(ACE) = 7) OR (AC.LOCATION(ACE) = 8)
      IF AC.TYPE(ACE) < 3
        IF NUM.OPEN.SPOTS > 0
          IF AVB.LAUNCH.TIME < AC.LAUNCH.TIME(ACE)
            LET LQSPOT = 3
          ELSE
            LET LQSPOT = 1
          ALWAYS
          FOR I = LQSPOT TO SPOT.AC(AC.TYPE(ACE),7),
            WITH SPOT(SPOT.AC(AC.TYPE(ACE),I)) = 0
            FIND THE FIRST CASE
            IF FOUND
              LET SPOTT = SPOT.AC(AC.TYPE(ACE),I)
              LET AC.DESTINATION(ACE) = SPOTT
              LET SPOT(SPOTT) = -AC.ID(ACE)
              SUBTRACT 1 FROM NUM.OPEN.SPOTS
              PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
              NUM.OPEN.SPOTS = **
              IF AC.SERVICE.FLAG(ACE) = 0
                IF TUG < 4
                  LET TUG = TUG + 1
                  LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
                  LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
                  LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
                  SCHEDULE AN AC.RESPOTTED GIVING ACE
                    IN TTRESPOT MINUTES
                ELSE
                  FILE ACE IN TUG.Q
            ALWAYS
          PRINT 1 LINE WITH ACE THUS
          AC ** FILED IN TUG.Q
        ALWAYS

```

```

ELSE **CHECK FOR FREE TUG AFTER SERVICE COMPLETE
ALWAYS
ELSE ** (IF NOT FOUND)
FILE THIS ACE IN SPOT.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

ALWAYS
ELSE ** (IF NUM.OPEN.SPOTS = 0)
FILE THIS ACE IN SPOT.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

ALWAYS
ELSE **ACE IS AV8
IF AC.SERVICE.FLAG(ACE) = 0
IF AC.FUEL.STAT(ACE) < 1
IF REFUELER < 4
ADD 1 TO REFUELER
LET DELAY = NORMAL.F(1.,.25,4)
SCHEDULE AN AC.REFUELED GIVING ACE
IN ((1. - AC.FUEL.STAT(ACE)) *
TTREFUEL.AC(AC.TYPE(ACE))) +
DELAY MINUTES
LET AC.SERVICE.FLAG(ACE) = 1
IF AC.LOAD.STAT(ACE) < 1
IF M.LOAD.SET(ACE) <> 1
IF M.LOAD.SET < 2
FILE THIS ACE IN LOAD.SET
PRINT 1 LINE WITH M.LOAD.SET THUS

N.LOAD.SET = **

LET XBAR = TTLOAD.AC(AC.TYPE(ACE))
LET SDEV = S.TTLOAD.AC(AC.TYPE(ACE))
LET TTLOAD = NORMAL.F(XBAR,SDEV,3)
+ (NUM.AV8S.LOADED / 4.)
ADD 1 TO NUM.AV8S.LOADED
SCHEDULE AN AC.LOADED GIVING ACE
IN TTLOAD MINUTES
LET AC.SERVICE.FLAG(ACE) = 1
ELSE
IF M.LOADER.Q(ACE) <> 1
FILE ACE IN LOADER.Q
ALWAYS
ALWAYS
ALWAYS
ELSE
FILE ACE IN REFUELER.Q
ALWAYS
ELSE
IF AC.LOAD.STAT(ACE) < 1
PRINT 1 LINE WITH AC.ID(ACE) THUS
AV8 ** FUELED, BUT NOT LOADED
IF M.LOAD.SET(ACE) <> 1
IF M.LOAD.SET < 2
FILE THIS ACE IN LOAD.SET
PRINT 1 LINE WITH M.LOAD.SET THUS

N.LOAD.SET = **

LET XBAR = TTLOAD.AC(AC.TYPE(ACE))
LET SDEV = S.TTLOAD.AC(AC.TYPE(ACE))
LET TTLOAD = NORMAL.F(XBAR,SDEV,3)
+ (NUM.AV8S.LOADED / 4.)

```

```

ADD 1 TO NUM.AV9S.LOADED
SCHEDULE AN AC.LOADED GIVING ACE
IN TTLOAD MINUTES
LET AC.SERVICE.FLAG(ACE) = 1
ELSE
  IF M.LOADER.Q(ACE) <> 1
    FILE ACE IN LOADER.Q
  ALWAYS
  ALWAYS
  ALWAYS
ELSE
  SCHEDULE AN AC.LOADED GIVING ACE NOW **AC AL
  **CONTR
  LET AC.SERVICE.FLAG(ACE) = 1
  ALWAYS
  ALWAYS
  ALWAYS
  ALWAYS
  ALWAYS

```

```

IF (AC.LOCATION(ACE) = 12)
  IF AC.TYPE(ACE) < 3
    IF NUM.OPEN.SPOTS > 0
      IF AV8.LAUNCH.TIME < AC.LAUNCH.TIME(ACE)
        LET LSPOT = 3
      ELSE
        LET LSPOT = 1
      ALWAYS
      FOR I = LSPOT TO SPOT.AC(AC.TYPE(ACE),7),
        WITH SPOT(SPOT.AC(AC.TYPE(ACE),I)) = 0
        FIND THE FIRST CASE
        IF FOUND
          LET SPOTT = SPOT.AC(AC.TYPE(ACE),I)
          LET AC.DESTINATION(ACE) = SPOTT
          LET SPOT(SPOTT) = -AC.ID(ACE)
          SUBTRACT 1 FROM NUM.OPEN.SPOTS
          PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
          NUM.OPEN.SPOTS = **
          IF SPOT(11) = 0
            LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
            LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
            LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
            SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
            IN TTARRIV.E MINUTES
            LET SPOT(11) = -AC.ID(ACE)
          ELSE
            IF M.ELEVATOR.Q(ACE) <> 1
              FILE ACE IN ELEVATOR.Q
            ALWAYS

```

```

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q

```

```

  ALWAYS
  ELSE ** (IF NOT FOUND)
    FILE THIS ACE IN SPOT.Q

```

```

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

```

```

  ALWAYS
  ELSE ** (IF NUM.OPEN.SPOTS = 0)

```

```

IF AC.TYPE(ACE) = 1
  FILE ACE IN BONE.FWD
  LET AC.DESTINATION(ACE) = 7
  IF SPOT(11) = 0
    LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
    LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
    LET TTARRIV.E = NORMAL.P(XBAR,SDEV,8)
    SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
      IN TTARRIV.E MINUTES
    LET SPOT(11) = -AC.ID(ACE)
  ELSE
    IF M.ELEVATOR.Q(ACE) <> 1
      FILE ACE IN ELEVATOR.Q
    ALWAYS

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q

    ALWAYS
    FILE ACE IN SPOT.Q
  ELSE
    FILE ACE IN BONE.AFT
    LET AC.DESTINATION(ACE) = 8
    IF SPOT(11) = 0
      LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
      LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
      LET TTARRIV.E = NORMAL.P(XBAR,SDEV,8)
      SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
        IN TTARRIV.E MINUTES
      LET SPOT(11) = -AC.ID(ACE)
    ELSE
      IF M.ELEVATOR.Q(ACE) <> 1
        FILE ACE IN ELEVATOR.Q
      ALWAYS

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q

    ALWAYS
    FILE ACE IN SPOT.Q
  ALWAYS

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

    ALWAYS
  ELSE
    **ACE IS AV9
    LET AC.DESTINATION(ACE) = 8
    FILE ACE IN BONE.AFT
    IF SPOT(11) = 0
      LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
      LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
      LET TTARRIV.E = NORMAL.P(XBAR,SDEV,8)
      SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
        IN TTARRIV.E MINUTES
      LET SPOT(11) = -AC.ID(ACE)
    ELSE
      IF ((M.ELEVATOR.Q(ACE) <> 1)
        AND (SPOT(11) <> -AC.ID(ACE)))
        FILE ACE IN ELEVATOR.Q
      ALWAYS

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q

    ALWAYS
  ALWAYS

```

```

        ALWAYS
    ELSE
        PRINT 1 LINE WITH COUNTER AND N.FLT.HAVE(FLIGHT) THUS
            COUNTER : **      N.FLT.HAVE : **

        ALWAYS
    ELSE **ACE NOT FOUND IN SHIP...
        PRINT 1 LINE WITH J,FLTARRAY(FLT.NUM(FLIGHT),J),
            FLT.TIME(FLIGHT) THUS
    **ERROR** THE *TH AC (***) OF THE FLIGHT TO LAUNCH AT **** IS NOT FOUND
    ALWAYS
    LOOP
    LET T.NUM.AC = FLT.AC.NUM(FLIGHT) - COUNTER
    LET TT.NUM.AC = T.NUM.AC
    PRINT 1 LINE WITH T.NUM.AC,FLT.AC.NUM(FLIGHT),COUNTER THUS
    T.NUM.AC = ** FLT.AC.NUM = ** COUNTER = **
    IF T.NUM.AC > 0 **CHECK DECK
        PRINT 1 LINE WITH T.NUM.AC THUS
        T.NUM.AC = ** 1 ...CHECKING DECK FOR AVAILABLE AC
        IF FLT.AC.TYPE(FLIGHT) <= 2 **ONLY HELOS CAN BE PARKED ON SPOTS
            FOR I = 1 TO 6
                WHILE T.NUM.AC > 0, DO
                    IF SPOT(I) <> 0
                        LET T.AC = ABS.P(SPOT(I))
                        FOR EACH ACE IN THE SHIP,
                            WITH AC.ID(ACE) = T.AC,
                                FIND THE FIRST CASE
                                    IF AC.TYPE(ACE) = FLT.AC.TYPE(FLIGHT)
                                        IF AC.LAUNCH.TIME(ACE) > (FLT.TIME(FLIGHT) + 20)
                                            IF AC.OP.STAT(ACE) >= .6
                                                IF AC.DESTINATION(ACE) = AC.LOCATION(ACE)
                                                    LET AC.LAUNCH.TIME(ACE) = FLT.TIME(FLIGHT)
                                                    IF AC.FUEL.STAT(ACE) < 1.0
                                                        IF REFUELER < 4
                                                            ADD 1 TO REFUELER
                                                            LET DELAY = NORMAL.P(1.,.25,4)
                                                            SCHEDULE AN AC.REFUELED GIVING ACE
                                                                IN ((1. - AC.FUEL.STAT(ACE)) *
                                                                    TTREFUEL.AC(AC.TYPE(ACE)))
                                                                + DELAY MINUTES
                                                            LET AC.SERVICE.FLAG(ACE) = 1
                                                        ELSE
                                                            FILE ACE IN REFUELER.3
                                                    ALWAYS
                                                ELSE
                                                    LET TTLOAD = 1. + GET1.P(1.5,3.0,3)
                                                                * TTLOAD.AC(AC.TYPE(ACE))
                                                    IF (AC.LAUNCH.TIME(ACE) - TIME.V) < 10
                                                        SCHEDULE AN AC.LOADED GIVING ACE
                                                            IN TTLOAD MINUTES
                                                    ELSE
                                                        SCHEDULE AN AC.LOADED GIVING ACE
                                                            AT AC.LAUNCH.TIME(ACE)
                                                            - 10. + TTLOAD
                                                    ALWAYS
                                                    LET AC.SERVICE.FLAG(ACE) = 1
                                                    ALWAYS
                                                    SUBTRACT 1 FROM T.NUM.AC

```

```

                                FILE THIS ACE IN FLT.WAVE(FLIGHT)
                                PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),
                                  AC.FUEL.STAT(ACE),AC.LOAD.STAT(ACE),
                                  FLT.TIME(FLIGHT),T.NUM.AC THUS
AC ** AT ** WITH FUEL ** AND LOAD ** JOINS FLIGHT **** ... T.NUM.AC = *
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                LOOP
                                ALWAYS
                                ALWAYS
                                IF T.NUM.AC > 0 **CHECK DELTA
                                PRINT 1 LINE WITH T.NUM.AC THUS
                                T.NUM.AC = ** 2 CHECKING DELTA FOR AVAILABLE AC
                                FOR EACH ACE IN DELTA.PATTERN
                                WHILE T.NUM.AC > 0, DO
                                IF AC.OP.STAT(ACE) > .6
                                IF AC.TYPE(ACE) = FLT.AC.TYPE(FLIGHT)
                                IF AC.LAUNCH.TIME(ACE) > (FLT.TIME(FLIGHT) + 20)
                                LET AC.LAUNCH.TIME(ACE) = FLT.TIME(FLIGHT)
                                SUBTRACT 1 FROM T.NUM.AC
                                FILE THIS ACE IN FLT.WAVE(FLIGHT)
                                PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),
                                  AC.DESTINATION(ACE),AC.FUEL.STAT(ACE),
                                  AC.LOAD.STAT(ACE),
                                  FLT.TIME(FLIGHT),T.NUM.AC THUS
AC ** AT ** WITH DEST ** FUEL ** LOAD ** JOINS FLIGHT **** ... T.NUM.AC = :
                                IF M.SPOT.Q(ACE) = 1
                                REMOVE THIS ACE FROM SPOT.Q
                                FILE THIS ACE IN SPOT.Q
                                PRINT 1 LINE WITH ACE THUS
                                AC ** RE-FILED IN SPOT.Q
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                LOOP
                                ALWAYS
                                **NO AC AVAIL FOR FLIGHT...LOOK IN BONE
                                IF T.NUM.AC > 0 **CHECK BONES
                                PRINT 1 LINE WITH T.NUM.AC THUS
                                T.NUM.AC = ** 3 CHECKING BONES FOR AVAILABLE AC
                                IF FLT.AC.TYPE(FLIGHT) = 1
                                FOR EACH ACE IN BONE.FWD
                                WHILE T.NUM.AC > 0, DO
                                IF AC.OP.STAT(ACE) >= .6
                                IF AC.TYPE(ACE) = FLT.AC.TYPE(FLIGHT)
                                IF AC.LAUNCH.TIME(ACE) > (FLT.TIME(FLIGHT) + 20)
                                LET AC.LAUNCH.TIME(ACE) = FLT.TIME(FLIGHT)
                                SUBTRACT 1 FROM T.NUM.AC
                                IF AC.TYPE(ACE) < 3
                                IF NUM.OPEN.SPOTS > 0
                                IF AVS.LAUNCH.TIME < AC.LAUNCH.TIME(ACE)
                                LET LOSPOT = 3
                                ELSE
                                LET LOSPOT = 1
                                ALWAYS
                                FOR I = LOSPOT TO SPOT.AC(AC.TYPE(ACE),7),

```

```

        WITH SPOT(SPOT.AC(AC.TYPE(ACE),I)) = 0
FIND THE FIRST CASE
IF FOUND
    LET SPOTT = SPOT.AC(AC.TYPE(ACE),I)
    LET AC.DESTINATION(ACE) = SPOTT
    LET SPOT(SPOTT) = -AC.IO(ACE)
    SUBTRACT 1 FROM NUM.OPEN.SPOTS
    IF TUG < 4
        LET TUG = TUG + 1
        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE AN AC.RESPOTTED GIVING ACE
            IN TTRESPOT MINUTES
    ELSE
        FILE ACE IN TUG.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q

        ALWAYS
    ELSE ** (IF NOT FOUND)
        FILE THIS ACE IN SPOT.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

        ALWAYS
    ELSE ** (IF NUM.OPEN.SPOTS = 0)
        FILE THIS ACE IN SPOT.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

        ALWAYS
    ELSE ** ACE IS AV8
        IF AC.FUEL.STAT(ACE) < 1
            IF REFUELER < 4
                ADD 1 TO REFUELER
                LET DELAY = NORMAL.F(1...25,4)
                SCHEDULE AN AC.REFUELED GIVING ACE
                    IN ((1. - AC.FUEL.STAT(ACE)) *
                        TTREFUEL.AC(AC.TYPE(ACE)))
                    + DELAY MINUTES
                LET AC.SERVICE.FLAG(ACE) = 1
                IF N.LOAD.SET(ACE) <> 1
                    IF AC.LOAD.STAT(ACE) < 1
                        IF N.LOAD.SET < 2
                            FILE ACE IN LOAD.SET
                            PRINT 1 LINE WITH N.LOAD.SET THUS

N.LOAD.SET = **

                            LET XBAR = TTLQAD.AC(AC.TYPE(ACE))
                            LET SDEV = S.TTLQAD.AC(AC.TYPE(ACE))
                            LET TTLQAD = NORMAL.F(XBAR,SDEV,3)
                                + (NUM.AV8S.LOADED / 4.)
                            ADD 1 TO NUM.AV8S.LOADED
                            SCHEDULE AN AC.LOADED GIVING ACE
                                IN TTLQAD MINUTES
                            LET AC.SERVICE.FLAG(ACE) = 1
                        ELSE
                            IF N.LOADER.3(ACE) <> 1
                                FILE ACE IN LOADER.Q
                            ALWAYS
                        ALWAYS
                    ALWAYS
                ALWAYS
            ALWAYS
        ALWAYS

```

```

ELSE
  FILE ACE IN REFUELER.Q
  ALWAYS
ELSE
  IF M.LOAD.SET(ACE) <> 1
  IF AC.LOAD.STAT(ACE) < 1
  IF M.LOAD.SET < 2
    FILE ACE IN LOAD.SET
    PRINT 1 LINE WITH M.LOAD.SET THUS

N.LOAD.SET = **

    LET XBAR = TTLOAD.AC(AC.TYPE(ACE))
    LET SDEV = S.TTLOAD.AC(AC.TYPE(ACE))
    LET TTLOAD = NORMAL.F(XBAR,SDEV,3)
      + (NUM.AV8S.LOADED / 4.)
    ADD 1 TO NUM.AV8S.LOADED
    SCHEDULE AN AC.LOADED GIVING ACE
      IN TTLOAD MINUTES
    LET AC.SERVICE.FLAG(ACE) = 1
  ELSE
    IF M.LOADER.Q(ACE) <> 1
    FILE ACE IN LOADER.Q
    ALWAYS
  ALWAYS
  ALWAYS
  ALWAYS
  ALWAYS
  FILE THIS ACE IN FLT.WAVE(FLIGHT)
  PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),
    AC.FUEL.STAT(ACE),AC.LOAD.STAT(ACE),
    FLT.TIME(FLIGHT),T.NUM.AC THUS
AC ** AT ** WITH FUEL ** AND LOAD ** JOINS FLIGHT *** ... T.NUM.AC = *
  ALWAYS
  ALWAYS
  ALWAYS
  LOOP
  ELSE
    FOR EACH ACE IN BONE.AFT
      WHILE T.NUM.AC > 0, DO
        IF AC.OP.STAT(ACE) >= .6
          IF AC.TYPE(ACE) = FLT.AC.TYPE(FLIGHT)
            IF AC.LAUNCH.TIME(ACE) > (FLT.TIME(FLIGHT) + 20)
              LET AC.LAUNCH.TIME(ACE) = FLT.TIME(FLIGHT)
              SUBTRACT 1 FROM T.NUM.AC
              IF NUM.OPEN.SPOTS > 0
                IF AV8.LAUNCH.TIME < AC.LAUNCH.TIME(ACE)
                  LET LOSPOT = 3
                ELSE
                  LET LOSPOT = 1
              ALWAYS
              FOR I = LOSPOT TO SPOT.AC(AC.TYPE(ACE),7),
                WITH SPOT(SPOT.AC(AC.TYPE(ACE),I)) = 0
              FIND THE FIRST CASE
              IF FOUND
                IF AC.TYPE(ACE) < 3
                  LET SPOTT = SPOT.AC(AC.TYPE(ACE),I)
                  LET AC.DESTINATION(ACE) = SPOTT
                  LET SPOT(SPOTT) = -AC.ID(ACE)
                  SUBTRACT 1 FROM NUM.OPEN.SPOTS
                  PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS

```

```

NUM.OPEN.SPOTS = **
IF TUG < 4
    LET TUG = TUG + 1
    LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
    LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
    LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
    SCHEDULE AN AC.RESPOTTED GIVING ACE
        IN TTRESPOT MINUTES
ELSE
    FILE ACE IN TUG.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q

    ALWAYS
ELSE
    **ACE IS AV8
    IF AC.FUEL.STAT(ACE) < 1
        IF REFUELER < 4
            ADD 1 TO REFUELER
            LET DELAY = NORMAL.F(1.,.25,4)
            SCHEDULE AN AC.REFUELED GIVING ACE
                IN ((1. - AC.FUEL.STAT(ACE)) *
                    TTREFUEL.AC(AC.TYPE(ACE)))
                    + DELAY MINUTES
            LET AC.SERVICE.FLAG(ACE) = 1
IF M.LOAD.SET(ACE) <> 1
    IF AC.LOAD.STAT(ACE) < 1
        IF N.LOAD.SET < 2
            FILE ACE IN LOAD.SET
            PRINT 1 LINE WITH N.LOAD.SET THUS

N.LOAD.SET = **

                LET XBAR = TTLLOAD.AC(AC.TYPE(ACE))
                LET SDEV = S.TTLLOAD.AC(AC.TYPE(ACE))
                LET TTLLOAD = NORMAL.F(XBAR,SDEV,3)
                    + (NUM.AV8S.LOADED / 4.)
                ADD 1 TO NUM.AV8S.LOADED
                SCHEDULE AN AC.LOADED GIVING ACE
                    IN TTLLOAD MINUTES
                LET AC.SERVICE.FLAG(ACE) = 1
            ELSE
                IF M.LOADER.Q(ACE) <> 1
                    FILE ACE IN LOADER.Q
                ALWAYS
            ALWAYS
        ALWAYS
    ALWAYS
ELSE
    FILE ACE IN REFUELER.Q
    ALWAYS
ELSE
    IF AC.LOAD.STAT(ACE) < 1
        IF M.LOAD.SET(ACE) <> 1
            IF N.LOAD.SET < 2
                FILE ACE IN LOAD.SET
                PRINT 1 LINE WITH N.LOAD.SET THUS

N.LOAD.SET= **

                LET XBAR = TTLLOAD.AC(AC.TYPE(ACE))
                LET SDEV = S.TTLLOAD.AC(AC.TYPE(ACE))
                LET TTLLOAD = NORMAL.F(XBAR,SDEV,3)
                    + (NUM.AV8S.LOADED / 4.)
                ADD 1 TO NUM.AV8S.LOADED
                SCHEDULE AN AC.LOADED GIVING ACE

```

```

                                IN TTLOAD MINUTES
                                LET AC.SERVICE.FLAG(ACE) = 1
                                ELSE
                                IF M.LOADER.Q(ACE) <> 1
                                FILE ACE IN LOADER.Q
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                ELSE ** (IF NOT FOUND)
                                FILE THIS ACE IN SPOT.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

                                ALWAYS
                                ELSE ** (IF NUM.OPEN.SPOTS = 0)
                                FILE THIS ACE IN SPOT.Q

PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q

                                ALWAYS
                                FILE THIS ACE IN FLT.WAVE(FLIGHT)
                                PRINT 1 LINE WITH AC.ID(ACE), AC.LOCATION(ACE),
                                AC.FUEL.STAT(ACE), AC.LOAD.STAT(ACE),
                                FLT.TIME(FLIGHT), T.NUM.AC THUS
AC ** AT ** WITH FUEL **. ** AND LOAD **. ** JOINS FLIGHT *** ... T.NUM.AC = *
                                ALWAYS
                                ALWAYS
                                ALWAYS
                                LOOP
                                ALWAYS
                                ALWAYS

IF TT.NUM.AC > 0
    LET TT.NUM.AC = TT.NUM.AC - T.NUM.AC
    ADD TT.NUM.AC TO NUM.REPLACED.AC
    ALWAYS

IF T.NUM.AC > 0
    PRINT 1 LINE WITH T.NUM.AC THUS
    T.NUM.AC = ** 4 ... UNABLE TO FIND AVAILABLE AC TO FILL THIS FLIGHT

    FOR I = 1 TO T.NUM.AC, DO
        FOR J = 1 TO FLT.AC.NUM(FLIGHT),
            WITH FLTARRAY(FLT.NUM(FLIGHT), J) <> 0
            FIND THE FIRST CASE
            IF FOUND
                FOR EACH ACE IN THE SHIP,
                    WITH AC.ID(ACE) = FLTARRAY(FLT.NUM(FLIGHT), J)
                    FIND THE FIRST CASE
                    IF FOUND
                        ADD 1 TO COUNTER
                        FILE THIS ACE IN FLT.WAVE(FLIGHT)
                        LET FLTARRAY(FLT.NUM(FLIGHT), J) = 0
                        PRINT 1 LINE WITH AC.ID(ACE), FLT.TIME(FLIGHT) THUS
NO REPLACEMENT AC ARE AVAILABLE... AC ** IS STILL SCHEDULED TO LAUNCH AT ***
                    ELSE
                        PRINT 1 LINE THUS
                        ***ERROR*** AC NOT FOUND IN FLTARRAY
                    ALWAYS

```

```

ELSE
  PRINT 1 LINE THUS
    ***ERROR*** AC NOT FOUND IN DECK.DECISION
  ALWAYS
  LOOP
  ALWAYS
  FILE FLIGHT IN PLAN
  SCHEDULE A FLIGHT.CHECK GIVING FLIGHT AT (FLT.TIME(FLIGHT)
    + (1. + BETA.F(1.5,3.0,9) * 2.))
  IF FLT.AC.TYPE(FLIGHT) = 3
    FILE FLIGHT IN AV8.PLAN
  ALWAYS
  IF N.SCHEDULE > 0
    REMOVE FIRST FLIGHT FROM SCHEDULE
    IF (FLT.TIME(FLIGHT) - TIME.V) >= 40
      SCHEDULE A DECK.DECISION GIVING FLIGHT IN
        (FLT.TIME(FLIGHT)-TIME.V-40) MINUTES
      FILE FLIGHT IN SCHEDULE
    ELSE
      SCHEDULE A DECK.DECISION GIVING FLIGHT IN 2 MINUTES
      FILE FLIGHT IN SCHEDULE
    ALWAYS
  ELSE
    SCHEDULE A STOP.SIMULATION IN 200 MINUTES
    PRINT 1 LINE WITH TIME.V,(TIME.V+200) THUS
    STOP.SIMULATION SCHEDULED AT ***,** TO OCCUR AT SIM.TIME = ***,**
  ALWAYS
  RETURN
END

```

```

** *****
EVENT DELTA.ARRIVAL GIVEN AC
** *****

DEFINE AC, ACE.T, FLAG, SPOTT AS INTEGER VARIABLES
DEFINE INTERVAL AS REAL VARIABLES

PRINT 1 LINE WITH AC.ID(AC),AC.FUEL.STAT(AC),AC.LOAD.STAT(AC),
AC.OP.STAT(AC),AC.FLYING.TIME(AC) THUS
AC ** WITH FUEL **.** AND LOAD **.** HAS OP.STAT **.** AND FLYING.TIME **.*

LET AC.DELTA.ARRIVAL.TIME(AC) = TIME.V
LET AC.LOCATION(AC) = 9
LET ETA(AC.ID(AC)) = 0

** THIS SECTION OF CODE UPDATES THE FUEL AND PRIORITY STATUS VARIABLES OF
** THE AIRCRAFT IN THE DELTA PATTERN

FOR EACH ACE IN DELTA.PATTERN, DO
LET DELTA = ((TIME.V - DELTA.UPDATE.TIME)/60) *
(FUELUSE.AC(AC.TYPE(ACE)))
LET AC.FUEL.STAT(ACE) = AC.FUEL.STAT(ACE) - DELTA
LET AC.FLYING.TIME(ACE) = (AC.FUEL.STAT(ACE)
/ FUELUSE.AC(AC.TYPE(ACE))) * 60.
LET AC.PRIORITY(ACE) = AC.PRIORITY(ACE) + DELTA * .2
PRINT 1 LINE WITH AC.ID(ACE),AC.DESTINATION(ACE),AC.PRIORITY(ACE),
AC.FUEL.STAT(ACE), AC.FLYING.TIME(ACE) THUS
AC ** WITH DEST ** HAS PRIORITY **.* FUEL.STAT **.** FLYING.TIME **.*
REMOVE THIS ACE FROM THE DELTA.PATTERN
FILE THIS ACE IN DELTA.PATTERN
LOOP
LET DELTA.UPDATE.TIME = TIME.V

** FILE THE ARRIVING AIRCRAFT IN THE DELTA PATTERN.

FILE THIS AC IN DELTA.PATTERN
FILE THIS AC IN SPOT.Q
PRINT 1 LINE WITH AC THUS
AC ** FILED IN SPOT.Q

** THIS SECTION OF CODE DETERMINES WHAT ACTION SHOULD BE TAKEN FOR THE
** AIRCRAFT IN DELTA. CHECK TO SEE
** IF THERE EXIST AN EMERGENCY STATUS FOR ANY OF THE AIRCRAFT IN THE DELTA
** PATTERN. IF AN EMERGENCY EXIST, ALLOW THE EMERGENCY AIRCRAFT TO RECOVER

LET FLAG = 0
FOR EACH ACE IN DELTA.PATTERN,
WITH ((AC.DESTINATION(ACE) = 9)
AND (AC.PRIORITY(ACE) > PRIORITY.STAT.AC(AC.TYPE(ACE))))
FIND THE FIRST CASE
IF FOUND
IF AC.PRIORITY(ACE) > EMERGENCY.STAT.AC(AC.TYPE(ACE))
SCHEDULE A SPOT.EMERGENCY GIVING ACE IN .25 MINUTES
PRINT 1 LINE WITH AC.ID(ACE),TIME.V THUS
AC ** DECLARES EMERGENCY AT **.*
LET FLAG = 1
ELSE
IF AC.PRIORITY(ACE) > PRIORITY.STAT.AC(AC.TYPE(ACE))
SCHEDULE A SPOT.PRIORITY GIVING ACE IN 1 MINUTE
PRINT 1 LINE WITH AC.ID(ACE),TIME.V THUS

```

```

AC ** DECLARES PRIORITY AT ****.**
    LET FLAG = 1
    ALWAYS
    ALWAYS
    ALWAYS
    IF FLAG = 0
        LET INTERVAL = 0
        LET I = 7

    PRINT 1 LINE THUS
    THE FOLLOWING AC ARE IN SPOT.Q:

    FOR EACH ACE IN SPOT.Q, DO
        IF AC.LOCATION(ACE) = 9
            LET INDEX(AC.ID(ACE)) = 8
        ELSE
            LET INDEX(AC.ID(ACE)) = 7
        ALWAYS
        PRINT 1 LINE WITH AC.ID(ACE), AC.LOCATION(ACE),
            AC.LAUNCH.TIME(ACE) THUS
        AC ** AT * SCHED TO LAUNCH AT ****
    LOOP

    WHILE ((NUM.OPEN.SPOTS > 0) AND (N.SPOT.Q > 0) AND (I > 1)), DO
        SUBTRACT 1 FROM I

    PRINT 1 LINE WITH NUM.OPEN.SPOTS, N.SPOT.Q, I, SPOT(I) THUS
    #OPEN SPOTS= ** #SPOT.Q = ** I = * SPOT(I) = **

    IF SPOT(I) = 0
        LET SPOTT = I
        IF N.AV8.PLAN > 0
            REMOVE THE FIRST FLIGHTS FROM AV8.PLAN
            LET AV8.LAUNCH.TIME = FLT.TIME(FLIGHTS)
            FILE THIS FLIGHTS IN AV8.PLAN
        ELSE
            LET AV8.LAUNCH.TIME = 9999
        ALWAYS

        FOR EACH FLIGHTS IN THE PLAN
            WITH ((FLT.AC.TYPE(FLIGHTS) <> 3)
                AND (FLT.DELAY(FLIGHTS) = 0))
            FIND THE FIRST CASE
            IF FOUND
                LET HELD.LAUNCH.TIME = FLT.TIME(FLIGHTS)
            ELSE
                LET HELD.LAUNCH.TIME = 9999
            ALWAYS

        PRINT 1 LINE WITH AV8.LAUNCH.TIME, HELD.LAUNCH.TIME,
            SPOTT THUS
    AV8.LAUNCH.TIME= **** HELD.LAUNCH.TIME= **** OPEN SPOT IS **

    FOR EACH ACE IN SPOT.Q,
        FOR J = 1 TO SPOT.AC(AC.TYPE(ACE), INDEX(AC.ID(ACE))),
            WITH ((SPOT.AC(AC.TYPE(ACE), J) = SPOTT)
                AND ((AC.LOCATION(ACE) = 9)
                    AND ((AC.FLYING.TIME(ACE) < 33)
                        OR (((AC.LAUNCH.TIME(ACE) - TIME.V)
                            > AC.FLYING.TIME(ACE))

```

```

OR ((AC.LAUNCH.TIME(ACE)-TIME.V) < 25))
AND (((AV8.LAUNCH.TIME
    > AC.LAUNCH.TIME(ACE))
OR (AV8.LAUNCH.TIME > (TIME.V + 20)))
OR ((AC.TYPE(ACE) = 3)
    AND (AV8.LAUNCH.TIME > (TIME.V + 10))))
AND ((HELO.LAUNCH.TIME > (TIME.V + 10))
    OR ((HELO.LAUNCH.TIME + 1.)
        > AC.LAUNCH.TIME(ACE))))))
OR ((AC.LOCATION(ACE) < 9)
    AND ((SPOTT > 2)
        OR (AV8.LAUNCH.TIME >= AC.LAUNCH.TIME(ACE)))
    AND ((AC.LAUNCH.TIME(ACE) - TIME.V) < 25)))
FIND THE FIRST CASE
IF FOUND ** AC COMPATIBLE WITH SPOT IDENTIFIED
PRINT 1 LINE WITH AC.ID(ACE),SPOTT,AC.LAUNCH.TIME(ACE),
    AC.FLYING.TIME(ACE) THUS
AC ** COMPATIBLE TO GO TO SPOT * HAS LAUNCH.TIME= **** AND FLYING.TIME = *

```

```

IF AC.LOCATION(ACE) = 9

```

```

IF FLAG = 0 **DRAWS TTRECOVER ONLY ON FIRST PASS
LET XBAR = TTRECOVER.AC(AC.TYPE(ACE))
LET SDEV = S.TTRECOVER.AC(AC.TYPE(ACE))
LET TTRECOVER = NORMAL.F(XBAR,SDEV,1)
IF TTRECOVER < (LAST.RECOVERY.TIME + .5 - TIME.V)
    LET TTRECOVER = LAST.RECOVERY.TIME + .5 - TIME.V
ALWAYS
IF TTRECOVER < (LAST.LAUNCH.TIME + 2 - TIME.V)
    LET TTRECOVER = LAST.LAUNCH.TIME + 2 - TIME.V
ALWAYS
LET FLAG = 1
ALWAYS

```

```

LET AC.DESTINATION(ACE) = SPOTT
LET SPOT(SPOTT) = -AC.ID(ACE)
SUBTRACT 1 FROM NUM.OPEN.SPOTS
PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
NUM.OPEN.SPOTS = **
SCHEDULE AN AC.RECOVERED GIVING ACE
    IN TTRECOVER + INTERVAL MINUTES
ADD UNIFORM.F(.5,.8,1) TO INTERVAL
LET LAST.RECOVERY.TIME = TTRECOVER + TIME.V
    + INTERVAL
PRINT 1 LINE WITH AC.ID(ACE),SPOTT,
    (TTRECOVER+INTERVAL) THUS

```

```

AC ** WILL RECOVER TO SPOT ** IN ***. * MINUTES
REMOVE THIS ACE FROM SPOT.Q

```

```

ELSE
IF AC.LOCATION(ACE) < 12
IF AC.TYPE(ACE) < 3

```

```

    LET AC.DESTINATION(ACE) = SPOTT
    LET SPOT(SPOTT) = -AC.ID(ACE)
    IF AC.SERVICE.FLAG(ACE) = 0
      IF TUG < 4
        ADD 1 TO TUG
        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE AN AC.RESPOTTED GIVING ACE
          IN TTRESPOT MINUTES
        PRINT 1 LINE WITH AC.ID(ACE),SPOTT THUS
        AC ** WILL RESPOT TO SPOT **
      ELSE
        FILE THIS ACE IN TUG.Q
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q
    ALWAYS
    ALWAYS
    IF AC.LOCATION(ACE) > 6
      SUBTRACT 1 FROM NUM.OPEN.SPOTS
      PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
      NUM.OPEN.SPOTS = **
    ALWAYS
    REMOVE THIS ACE FROM SPOT.Q
  ELSE
    IF AC.LAUNCH.TIME(ACE) = AV8.LAUNCH.TIME
    IF SPOTT <= 2 **AV8'S CAN RECOVER ON 1,2,5,6 BUT LA
    IF (AC.LAUNCH.TIME(ACE) - TIME.V) <= 10.
      LET AC.DESTINATION(ACE) = SPOTT
      LET SPOT(SPOTT) = -AC.ID(ACE)
      LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
      LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
      LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
      SCHEDULE AN AC.RESPOTTED GIVING ACE IN
        TTRESPOT MINUTES
      PRINT 1 LINE WITH AC.ID(ACE),SPOTT THUS
    AC = ** WILL RESPOT TO **
      SUBTRACT 1 FROM NUM.OPEN.SPOTS
      PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
      NUM.OPEN.SPOTS = **
      REMOVE THIS ACE FROM SPOT.Q
    ELSE
      SCHEDULE A SPOT.OPEN IN (AC.LAUNCH.TIME(ACE)
        - 10. + UNIFORM.F(.1,2,.9)) MINUTES
    ALWAYS
    ALWAYS
    ALWAYS
    ALWAYS
  ELSE
    IF AC.TYPE(ACE) < 3
      LET AC.DESTINATION(ACE) = SPOTT
      LET SPOT(SPOTT) = - AC.ID(ACE)
      SUBTRACT 1 FROM NUM.OPEN.SPOTS
      PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
      NUM.OPEN.SPOTS = **
    ELSE
      LET AC.DESTINATION(ACE) = 3
      FILE ACE IN BONE.AFT
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN BONE

```

```

      ALWAYS
      IF (SPOT(11) = 0) AND
        (HANGER.EQUIV < MAX.HANGER.EQUIV)
        LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
        LET TTARRIV.E = NORMAL.P(XBAR,SDEV,0)
        SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
          IN TTARRIV.E MINUTES
        LET SPOT(11) = -AC.ID(ACE)
      ELSE
        IF N.ELEVATOR.Q(ACE) <> 1
          FILE ACE IN ELEVATOR.Q
          ALWAYS
      PRINT 1 LINE WITH ACE THUS
      AC ** FILED IN ELEVATOR.Q
        ALWAYS
        REMOVE ACE FROM SPOT.Q
        ALWAYS
        ALWAYS
        PRINT 1 LINE WITH N.SPOT.Q,NUM.OPEN.SPOTS,I,SPOT(I),
          AC.ID(ACE), AC.LOCATION(ACE) THUS
      #IN SPOT.Q ** #OPEN SPOTS * I * SPOT(I) ** AC.ID ** AC.LOC **
        ALWAYS
        ALWAYS
      LOOP
      ALWAYS
      RETURN
      END

```

```

** *****
EVENT DELTA.UPDATE.CHK
** *****

PRINT 1 LINE WITH DELTA.UPDATE.TIME, (TIME.V-DELTA.UPDATE.TIME) THUS
DELTA.UPDATE.TIME= ****.****    TIME-DELTA= ****.****
IF (TIME.V - DELTA.UPDATE.TIME) >= 5
    SCHEDULE A SPOT.OPEN NOW
    SCHEDULE A DELTA.UPDATE.CHK IN 5 MINUTES
ELSE
    SCHEDULE A DELTA.UPDATE.CHK IN
        MAX.P(5.-TIME.V+DELTA.UPDATE.TIME, 1.0) MINUTES
ALWAYS
RETURN
END

```

```

** *****
EVENT ELEVATOR.ARRIVAL GIVEN AC
** *****

```

```

DEFINE AC AS AN INTEGER VARIABLE

```

```

IF AC.LOCATION(AC) < 12
  LET TUG = TUG - 1  **ALL AIRCRAFT ARE MOVED BY TUG TO AND FROM THE ELEVATOR
  IF AC.LOCATION(AC) < 7
    LET SPOT(AC.LOCATION(AC)) = 0
    ADD 1 TO NUM.OPEN.SPOTS
    PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
    NUM.OPEN.SPOTS = **
    SCHEDULE A SPOT.OPEN NOW
  ELSE
    IF AC.LOCATION(AC) = 7
      REMOVE AC FROM BONE.FWD
    ELSE
      IF AC.LOCATION(AC) = 8
        REMOVE AC FROM BONE.AFT
      ALWAYS
    ALWAYS
  ALWAYS
  LET TTARRIV.H = .5 + ((BETA.F(1.5,5.0,7)) * 4.)
  SCHEDULE A HANGER.ARRIVAL GIVING AC IN TTARRIV.H MINUTES
  LET AC.LOCATION(AC) = 11
  LET SPOT(11) = AC.ID(AC)
  IF (N.TUG.Q > 0) AND (TUG < 4)
    FOR EACH ACE IN TUG.Q
      WITH (AC.DESTINATION(ACE) < 9) AND (AC.LOCATION(ACE) <> 12)
      FIND THE FIRST CASE
      IF FOUND
        REMOVE THIS ACE FROM TUG.Q
        PRINT 1 LINE WITH ACE THUS
AC ** REMOVED FROM TUG.Q
      LET TUG = TUG + 1
      IF AC.DESTINATION(ACE) = 7 OR AC.DESTINATION(ACE) = 8
        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE A BONE.ARRIVAL GIVING ACE IN TTRESPOT MINUTES
      ELSE
        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE AN AC.RESPOTTED GIVING ACE IN TTRESPOT MINUTES
      ALWAYS
    ALWAYS
  ALWAYS
  ELSE **AC IS GOING TO THE DECK
    PRINT 1 LINE THUS
AC GOING TO DECK HAS ARRIVED AT THE ELEVATOR
    REMOVE THIS AC FROM THE HANGER.DECK
    LET TTARRIV.D = .5 + ((BETA.F(1.5,5.0,7)) * 4.)
    SCHEDULE A DECK.ARRIVAL GIVING AC IN TTARRIV.D MINUTES
  ALWAYS
  RETURN
END

```

```

*****
EVENT FLIGHT.CHECK GIVEN FLIGHT
*****
DEFINE FLIGHT AS AN INTEGER VARIABLE
DEFINE DEL.T.CMK AS REAL VARIABLES

IF N.PLAN(FLIGHT) = 1 **FLIGHT HAS NOT LAUNCHED
  IF FLT.AC.NUM(FLIGHT) > 0
    FOR EACH ACE IN FLT.WAVE(FLIGHT), DO
      PRINT 1 LINE WITH AC.ID(ACE), AC.LOCATION(ACE), AC.FUEL.STAT(ACE),
        AC.LOAD.STAT(ACE), AC.OP.STAT(ACE),
        AC.LAUNCH.TIME(ACE), AC.DESTINATION(ACE) THUS
AC ** AT ** WITH FUEL ** LOAD ** HAS OP.STAT ** TO.TIME **** DEST **

      IF ((AC.LOCATION(ACE) < 7) AND (AC.DESTINATION(ACE) = 10))
        IF AC.OP.STAT(ACE) = 1.0
          FILE ACE IN AC.RDY.SET
        ELSE
          FILE ACE IN AC.PRE.RDY.SET
        ALWAYS
      ELSE
        FILE ACE IN AC.NOT.RDY.SET
      ALWAYS
    LOOP
    PRINT 1 LINE WITH N.AC.RDY.SET, N.AC.PRE.RDY.SET,
      N.AC.NOT.RDY.SET THUS
#RDY AC= ** #PRE.RDY AC= ** #NOT.RDY AC= **

    FOR EACH FLIGHTS IN THE PLAN, DO
      PRINT 1 LINE WITH FLT.TIME(FLIGHTS), FLT.AC.TYPE(FLIGHTS),
        FLT.AC.RDY(FLIGHTS), FLT.AC.NUM(FLIGHTS),
        FLT.DELAY(FLIGHTS) THUS
FLT SCHED AT **** WITH AC.TYPE * HAS * OUT OF * READY...#DELAYS= *
    LOOP

    IF FLT.AC.TYPE(FLIGHT) <> 3 **AV-8S ALWAYS FLY IN TWO'S
      IF N.AC.RDY.SET <> FLT.AC.NUM(FLIGHT)
        IF (N.AC.RDY.SET+N.AC.PRE.RDY.SET) <> FLT.AC.NUM(FLIGHT)
          ADD 1 TO SPLIT.FLTS
          LET T.FLT = SPLIT.FLTS + NUM.FLTS
          CREATE A FLIGHTS CALLED F(T.FLT)

      IF N.SET.TEMP > 0
        FOR EACH ACE IN SET.TEMP, DO
          REMOVE THIS ACE FROM SET.TEMP
        LOOP
      ALWAYS

      FOR EACH ACE IN FLT.WAVE(FLIGHT), DO
        FILE ACE IN SET.TEMP
        REMOVE THIS ACE FROM FLT.WAVE(FLIGHT)
        PRINT 1 LINE WITH AC.ID(ACE) THUS
        AC ** FILES IN SET.TEMP
      LOOP

      FOR EACH ACE IN SET.TEMP, DO
        IF AC.LOCATION(ACE) > 6
          FILE THIS ACE IN FLT.WAVE(F(T.FLT))
          PRINT 1 LINE WITH AC.ID(ACE),

```

```

                                (FLT.TIME(FLIGHT)+5) THUS
AC *** IS NOW SCHEDULED TO LAUNCH AT ***
ELSE
    FILE THIS ACE IN FLT.WAVE(FLIGHT)
    PRINT 1 LINE WITH AC.ID(ACE),
                                (FLT.TIME(FLIGHT)) THUS
AC *** SCHEDULED TO LAUNCH AT ***
ALWAYS
LOOP

LET FLT.NUM(FCT.FLT)) = FLT.NUM(FLIGHT) + .1
LET FLT.DELAY(FCT.FLT)) = FLT.DELAY(FLIGHT) + 5
LET FLT.TIME(FCT.FLT)) = FLT.TIME(FLIGHT) + 5
WHILE FLT.TIME(FCT.FLT)) < TIME.V, DO
    ADD 5 TO FLT.TIME(FCT.FLT))
    ADD 5 TO FLT.DELAY(FCT.FLT))
    ADD .1 TO FLT.NUM(FCT.FLT))
LOOP
LET FLT.AC.TYPE(FCT.FLT)) = FLT.AC.TYPE(FLIGHT)
LET FLT.AC.NUM(FCT.FLT)) = N.FLT.WAVE(FCT.FLT))
LET FLT.AC.RDY(FCT.FLT)) = 0
LET FLT.AC.NUM(FLIGHT) = N.FLT.WAVE(FLIGHT)
PRINT 1 LINE WITH FLT.AC.NUM(FLIGHT), FLT.AC.RDY(FLIGHT) THUS
FLIGHT HAS *** AC, OF WHICH *** ARE READY TO LAUNCH
IF ((FLT.AC.RDY(FLIGHT) = FLT.AC.NUM(FLIGHT))
    AND (FLT.AC.NUM(FLIGHT) > 0))
    SCHEDULE A FLIGHT.LAUNCH GIVING FLIGHT NOW
    CALL CHECK1 GIVING FLIGHT
ELSE
    IF FLT.AC.NUM(FLIGHT) = 0
        REMOVE FLIGHT FROM PLAN
    ALWAYS
ALWAYS

LET DEL = (FLT.NUM(FCT.FLT)) - INT.F(FLT.NUM(FCT.FLT)))
PRINT 1 LINE WITH FLT.NUM(FCT.FLT),
                                INT.F(FLT.NUM(FCT.FLT)), DEL THUS
FLT.NUM = ***.* INT(FLT.NUM) = ***.* DEL = ***.*
IF DEL < .4
    FILE FCT.FLT) IN PLAN
    LET T.CHR=(FLT.TIME(FCT.FLT))+(1.+9ETA.F(1.5,3.0,9)*2.))
    PRINT 1 LINE WITH T.CHR THUS
FCT.FLT) SCHEDULED FOR FLIGHT.CHECK AT ***.*
    SCHEDULE A FLIGHT.CHECK GIVING FCT.FLT)) AT T.CHR
ELSE **AFTER 4 FLIGHT.CHECKS THERE ARE AC IN THIS FLIGHT STILL N
    FOR EACH ACE IN FLT.WAVE(FCT.FLT)), DO
        ADD 1 TO NUM.CANCELLED.MISSIONS
        FOR EACH FLIGHT IN THE SCHEDULE
            FOR I = 1 TO FLT.AC.NUM(FLIGHT)
                WITH AC.ID(ACE) = FLT.ARRAY(FLT.NUM(FLIGHT),I)
                FIND THE FIRST CASE
                IF FOUND
                    LET AC.LAUNCH.TIME(ACE) = FLT.TIME(FLIGHT)
                ELSE
                    LET AC.LAUNCH.TIME(ACE) = 9999
            ALWAYS
        IF ((AC.LOCATION(ACE) < 9) AND (M.SPOT.Q(ACE) = 1))
            REMOVE THIS ACE FROM SPOT.Q
        ALWAYS

```

```

IF M.BONE.FWD(ACE) = 1
  REMOVE THIS ACE FROM BONE.FWD
  FILE THIS ACE IN BONE.FWD
  ALWAYS

IF M.BONE.AFT(ACE) = 1
  REMOVE THIS ACE FROM BONE.AFT
  FILE THIS ACE IN BONE.AFT
  ALWAYS

IF ((AC.DESTINATION(ACE) < 7) **NEED TO CANCEL RESPOT
  AND (AC.LOCATION(ACE) < 9)) **FOR AC ON DECK
  FOR EACH AC.RESPOTTED IN EV.S(I.AC.RESPOTTED)
    WITH AC3 = ACE, DO
      CANCEL AC.RESPOTTED
      PRINT 1 LINE WITH AC.ID(ACE), TIME.V THUS
AC.RESPOTTED EVENT FOR AC ** WAS CANCELLED AT ****.
    LOOP
    LET SPOT(AC.DESTINATION(ACE)) = 0
    ADD 1 TO NUM.OPEN.SPOTS
    SCHEDULE A SPOT.OPEN NOW
  ALWAYS

      PRINT 1 LINE WITH AC.ID(ACE), AC.LAUNCH.TIME(ACE) THUS
AC **'S STICK IN THIS FLIGHT IS CANCELLED...NEXT LAUNCH.TIME IS ****
    LOOP
    FOR EACH FLIGHT IN PLAN
      WITH (FLT.TIME(FLIGHT) = FLT.TIME(FLIGHT))
      AND (FLT.NUM(FLIGHT) = FLT.NUM(FLIGHT))
    FIND THE FIRST CASE
    IF FOUND
      REMOVE THIS FLIGHT FROM PLAN
    ELSE
      PRINT 1 LINE THUS
      **ERROR** FLIGHT NOT FOUND IN PLAN
    ALWAYS
    IF ((N.PLAN = 0) AND (N.SCHEDULE = 0))
      SCHEDULE A STOP.SIMULATION IN 160 MINUTES
      PRINT 1 LINE WITH TIME.V,(TIME.V+160) THUS
AT ****. A STOP SIMULATION WAS SCHEDULED FOR TIME = ****.
    ALWAYS
  ALWAYS

  SCHEDULE A FLIGHT.CHECK GIVING FLIGHT
  IN (1. + BETA.F(1.5,3.0,9) * 2.) MINUTES
  ELSE **ALL AC ARE IN FINAL SERVICE...WAIT
  PRINT 1 LINE THUS
  ALL AC ARE IN FINAL SERVICE ON DECK SPOTS...WAIT FOR COMPLETION
  SCHEDULE A FLIGHT.CHECK GIVING FLIGHT
  IN (1. + BETA.F(1.5,3.0,9) * 2.) MINUTES
  ALWAYS
  ELSE **ALL AC ARE READY...LAUNCH IS IMMINENT
  PRINT 1 LINE THUS
  ALL AC ARE READY...LAUNCH IS IMMINENT
  SCHEDULE A FLIGHT.CHECK GIVING FLIGHT
  IN (1. + BETA.F(1.5,3.0,9) * 2.) MINUTES
  ALWAYS
  ELSE **AV3 FLT IS LATE

```

```

PRINT 1 LINE THUS
THIS AV8 *LIGHT IS LATE
ADD 2 TO FLT.DELAY(FLIGHT)
IF((N.AC.RDY.SET <> FLT.AC.NUM(FLIGHT))
OR (FLT.DELAY(FLIGHT) > 14))
IF FLT.DELAY(FLIGHT) >= 10 **CANCEL THIS FLIGHT
ADD 2 TO NUM.CANCELLED.MISSIONS
FOR EACH ACE IN FLT.WAVE(FLIGHT), DO
FOR EACH FLIGHT IN THE SCHEDULE
FOR I = 1 TO FLT.AC.NUM(FLIGHT)
WITH AC.ID(ACE) = FLTARRAY(FLT.NUM(FLIGHT),I)
FIND THE FIRST CASE
IF FOUND
LET AC.LAUNCH.TIME(ACE) = FLT.TIME(FLIGHT)
ELSE
LET AC.LAUNCH.TIME(ACE) = 9999
ALWAYS

IF ((AC.LOCATION(ACE) < 9) AND (M.SPOT.Q(ACE) = 1))
REMOVE THIS ACE FROM SPOT.Q
ALWAYS

IF M.BONE.AFT(ACE) = 1
REMOVE THIS ACE FROM BONE.AFT
FILE THIS ACE IN BONE.AFT
ALWAYS

IF ((AC.DESTINATION(ACE) < 7) **NEED TO CANCEL RESPOT
AND (AC.LOCATION(ACE) < 9)) **FOR AC ON DECK
FOR EACH AC.RESPOTTED IN EV.S(I,AC.RESPOTTED)
WITH AC3 = ACE, DO
CANCEL AC.RESPOTTED
PRINT 1 LINE WITH AC.ID(ACE), TIME.V THUS
AC.RESPOTTED EVENT FOR AC ** WAS CANCELLED AT ****.*
LOOP
LET SPOT(AC.DESTINATION(ACE)) = 0
ADD 1 TO NUM.OPEN.SPOTS
SCHEDULE A SPOT.OPEN NOW
ALWAYS

IF AC.LOCATION(ACE) < 7 **NEED TO RESPOT AV8S BACK TO BONE
SUBTRACT 1 FROM NUM.AV8.RDY
LET AC.DESTINATION(ACE) = 8
LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
SCHEDULE A BONE.ARRIVAL GIVING ACE
IN TTRESPOT MINUTES
FILE ACE IN BONE.AFT
ALWAYS

PRINT 1 LINE WITH AC.ID(ACE), AC.LAUNCH.TIME(ACE) THUS
AC **'S STICK IN THIS FLIGHT IS CANCELLED...NEXT LAUNCH.TIME IS ****
LOOP
FOR EACH FLIGHT IN PLAN
WITH (FLT.TIME(FLIGHT) = FLT.TIME(FLIGHT))
AND (FLT.NUM(FLIGHT) = FLT.NUM(FLIGHT))
FIND THE FIRST CASE
IF FOUND
REMOVE THIS FLIGHT FROM PLAN

```

```

        REMOVE THIS FLIGHT FROM AV8.PLAN
    ELSE
        PRINT 1 LINE THUS
        **ERROR** FLIGHT NOT FOUND IN PLAN
    ALWAYS
    IF ((N.PLAN = 0) AND (N.SCHEDULE = 0))
        SCHEDULE A STOP.SIMULATION IN 160 MINUTES
        PRINT 1 LINE WITH TIME.V,(TIME.V+160) THUS
    AT ***.** A STOP SIMULATION WAS SCHEDULED FOR TIME = ***.**
    ALWAYS
    ELSE
        SCHEDULE A FLIGHT.CHECK GIVING FLIGHT IN 2 MINUTES
    ALWAYS
    ELSE
        SCHEDULE A FLIGHT.CHECK GIVING FLIGHT IN 2 MINUTES
        PRINT 1 LINE THUS
    ALL AV8-'S ARE ON DECK AND READY TO LAUNCH...
    ALWAYS
    ALWAYS

    FOR EACH ACE IN AC.RDY.SET, DO
        REMOVE ACE FROM AC.RDY.SET
    LOOP
    FOR EACH ACE IN AC.PRE.RDY.SET, DO
        REMOVE ACE FROM AC.PRE.RDY.SET
    LOOP
    FOR EACH ACE IN AC.NOT.RDY.SET, DO
        REMOVE ACE FROM AC.NOT.RDY.SET
    LOOP
    ELSE
        IF N.PLAN(FLIGHT) = 1
            REMOVE THIS FLIGHT FROM PLAN
        ALWAYS
    ALWAYS
    ELSE **FLIGHT HAS LAUNCHED
        PRINT 1 LINE THUS
    THIS FLIGHT HAS LAUNCHED
    ALWAYS

    RETURN
    END

```

```

*****
EVENT FLIGHT.LAUNCH GIVEN FLIGHT
*****

DEFINE INTERVAL, T.FLT.NUM AS REAL VARIABLES
DEFINE FLIGHT AS INTEGER VARIABLES

LET INTERVAL = UNIFORM.F(.5,1.,5)

PRINT 1 LINE WITH INTERVAL, LAST.LAUNCH.TIME, LAST.RECOVERY.TIME THUS
INTERVAL= **.* LAST.LAUNCH.TIME= **.*.* LAST.REC.TIME= **.*.*

IF TIME.V + INTERVAL < MAX.F((LAST.LAUNCH.TIME + 1.),
                             (LAST.RECOVERY.TIME + 1.))
    LET INTERVAL = MAX.F((LAST.LAUNCH.TIME - TIME.V + 1.),
                        (LAST.RECOVERY.TIME - TIME.V + 1.))
ALWAYS
LET T.FLT.NUM = INT.F(FLT.NUM(FLIGHT))
IF FLT.AC.TYPE(FLIGHT) = 3
    FOR EACH ACE IN FLT.WAVE(FLIGHT), DO
        IF AC.LOCATION(ACE) < 7
            PRINT 1 LINE WITH TIME.V, AC.ID(ACE), INTERVAL THUS
            TIME: **** AC ** WILL LAUNCH IN *.* MINUTES
            SCHEDULE AN AC.LAUNCHED GIVING ACE, T.FLT.NUM IN INTERVAL MINUTES
            LET LAST.LAUNCH.TIME = TIME.V + INTERVAL
            REMOVE THIS ACE FROM FLT.WAVE(FLIGHT)
            ADD UNIFORM.F(.3,.7,5) TO INTERVAL
        ELSE
            PRINT 1 LINE WITH AC.ID(ACE) THUS
AC ** WAS LAUNCHED EARLIER...DURING AN EMERGENCY RECOVERY...
        ALWAYS
    LOOP
    SCHEDULE A SPOT.OPEN IN INTERVAL MINUTES
    FOR EACH FLIGHT IN AV8.PLAN
        WITH FLT.TIME(FLIGHT) = FLT.TIME(FLIGHTS)
        FIND THE FIRST CASE
        IF FOUND
            REMOVE THIS FLIGHTS FROM AV8.PLAN
        ELSE
            PRINT 1 LINE THUS
            **ERROR** FLIGHT NOT FOUND IN PLAN (AV8)
        ALWAYS
    ELSE
        IF N.SET.TEMP > 0
            FOR EACH ACE IN SET.TEMP, DO
                REMOVE THIS ACE FROM SET.TEMP
            LOOP
        ALWAYS
        FOR EACH ACE IN FLT.WAVE(FLIGHT), DO
            FILE ACE IN SET.TEMP
            REMOVE THIS ACE FROM FLT.WAVE(FLIGHT)
        LOOP

        FOR EACH ACE IN SET.TEMP, DO
            FILE ACE IN FLT.WAVE(FLIGHT)
        LOOP

```

```

FOR EACH ACE IN FLT.WAVE(FLIGHT), DO
  IF AC.LOCATION(ACE) < 7
    PRINT 1 LINE WITH TIME.V, AC.ID(ACE), INTERVAL THUS
    TIME: **** AC ** WILL LAUNCH IN *.* MINUTES
    SCHEDULE AN AC.LAUNCHED GIVING ACE, T.FLT.NUM IN INTERVAL MINUTES
    LET LAST.LAUNCH.TIME = TIME.V + INTERVAL
    REMOVE THIS ACE FROM FLT.WAVE(FLIGHT)
    ADD UNIFORM.F(.3,.7,5) TO INTERVAL

    IF INTERVAL > 10.0
      PRINT 1 LINE WITH INTERVAL THUS
      **ERROR** INTERVAL = ***
      PRINT 1 LINE WITH N.SET.TEMP, N.FLT.WAVE(FLIGHT) THUS
      #IN SET.TEMP = ***** #IN FLT.WAVE = *****
      SCHEDULE A STOP.SIMULATION NOW
      RETURN
    ALWAYS

  ELSE
    PRINT 1 LINE WITH AC.ID(ACE) THUS
    AC ** WAS LAUNCHED EARLIER... DURING AN EMERGENCY RECOVERY...
    ALWAYS
    LOOP
    SCHEDULE A SPOT.OPEN IN INTERVAL MINUTES
    ALWAYS

  FOR EACH FLIGHTE IN PLAN
    WITH (FLT.TIME(FLIGHT) = FLT.TIME(FLIGHTE))
      AND (FLT.NUM(FLIGHT) = FLT.NUM(FLIGHTE))
    FIND THE FIRST CASE
    IF FOUND
      REMOVE THIS FLIGHTE FROM PLAN
    ELSE
      PRINT 1 LINE THUS
      **ERROR** FLIGHT NOT FOUND IN PLAN
    ALWAYS

  FOR EACH FLIGHTE IN THE PLAN, DO
    FOR EACH ACE IN FLT.WAVE(FLIGHTE), DO
      PRINT 1 LINE WITH FLT.TIME(FLIGHTE), AC.ID(ACE),
        AC.LOCATION(ACE), AC.DESTINATION(ACE) THUS
    FLIGHT **** IS IN PLAN AND HAS AC ** AT LOCATION ** WITH DEST **
    LOOP
  LOOP

  IF ((N.PLAN = 0) AND (N.SCHEDULE = 0))
    SCHEDULE A STOP.SIMULATION IN 160 MINUTES
    PRINT 1 LINE WITH TIME.V, (TIME.V+160) THUS
    AT ****.** A STOP SIMULATION WAS SCHEDULED FOR TIME = ****.**
  ALWAYS
  RETURN
END

```

```

*****
EVENT HANGER.ARRIVAL GIVEN AC
*****

DEFINE AC AS INTEGER VARIABLES
IF AC.TYPE(AC) = 2
    ADD 1.5 TO HANGER.EQUIV
ELSE
    ADD 1.0 TO HANGER.EQUIV
ALWAYS

PRINT 1 LINE WITH HANGER.EQUIV THUS
HANGER.EQUIV = **.*

FILE AC IN HANGER.DECK
PRINT 1 LINE WITH AC THUS
AC ** FILED IN HANGER
LET AC.LOCATION(AC) = 12
LET SPOT(11) = 0
IF N.ELEVATOR.Q > 0
    REMOVE THE FIRST ACE FROM ELEVATOR.Q
    IF AC.DESTINATION(ACE) = 12
        IF (HANGER.EQUIV + 1) < (MAX.HANGER.EQUIV)
            LET SPOT(11) = -AC.ID(ACE)
            IF TUG < 4
                ADD 1 TO TUG
                LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
                LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
                LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
                IN TTARRIV.E MINUTES
                PRINT 1 LINE WITH ACE,TTARRIV.E THUS
AC ** SCHEDULED TO ARRIVE AT THE ELEVATOR IN **.* MINUTES
            ELSE
                FILE ACE IN TUG.Q
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q
            ALWAYS
        ELSE
            IF N.ELEVATOR.Q(ACE) <> 1
                FILE ACE IN ELEVATOR.Q
            ALWAYS
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN ELEVATOR.Q
        FOR EACH ACE IN ELEVATOR.Q
            WITH AC.DESTINATION(ACE) < 12
                FIND THE FIRST CASE
                IF FOUND
                    REMOVE THIS ACE FROM ELEVATOR.Q
                    LET SPOT(11) = -AC.ID(ACE)
                    LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
                    LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
                    LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
                    SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
                    IN TTARRIV.E MINUTES
                    PRINT 1 LINE WITH ACE,TTARRIV.E THUS
ACE ** SCHEDULED TO ARRIVE AT THE ELEVATOR IN **.* MINUTES
                ALWAYS
            ALWAYS
        ELSE

```

```

      LET SPOT(11) = -AC.ID(ACE)
      LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
      LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
      LET TTARRIV.E = NORMAL.F(XBAR,SDEV,8)
      SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE IN TTARRIV.E MINUTES
      PRINT 1 LINE WITH ACE,TTARRIV.E THUS
AC ** SCHEDULED TO ARRIVE AT THE ELEVATOR IN **.* MINUTES
  ALWAYS
  ALWAYS
  RETURN
  END

```

```

*****
EVENT SPOT.EMERGENCY GIVEN AC
*****
DEFINE SPOTT, AC,ACE,MAX.LAUNCH.TIME,MIN.LAUNCH.TIME,AC.MIN,AC.ROY,
      ACE.P, ACE.T, FLIGHTE.P, FLAG, AND AC.MAX AS INTEGER VARIABLES
DEFINE INTERVAL, TTRECOVER, T.FLT.NUM AS REAL VARIABLES
PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC),AC.PRIORITY(AC),
      AC.FUEL.STAT(AC) THUS
T ****.E ** AC ** W PRI ** FUEL ** REQUESTED EMERGENCY LANDING

IF M.SPOT.Q(AC) = 1 ** OTHERWISE AC ALREADY HAS A RECOVERY DESTINATION

IF NUM.OPEN.SPOTS > 0
  FOR I BACK FROM 6 TO 1 BY 1
    WITH SPOT(I) = 0
    FIND THE FIRST CASE
    IF FOUND
      LET SPOTT = I
    ELSE
      LET SPOTT = 0
      PRINT 1 LINE THUS
**ERROR** OPEN SPOT NOT FOUND
  ALWAYS
  ALWAYS
  IF I <> 0
    PRINT 1 LINE WITH EVENT.V,NUM.OPEN.SPOTS,SPOTT THUS
EVENT ** #OPEN SPOTS ** OPEN SPOT * IF
  SUBTRACT 1 FROM NUM.OPEN.SPOTS

PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
NUM.OPEN.SPOTS = **
ELSE
  LET FLAG = 0
  FOR I BACK FROM 6 TO 1 BY 1
    WHILE FLAG = 0, 00
      LET SPOTT = I
      IF SPOT(SPOTT) < 0
        FOR EACH ACE IN THE SHIP
          WITH AC.ID(ACE) = SPOT(SPOTT)
          FIND THE FIRST CASE
          IF FOUND
            PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),
              AC.FUEL.STAT(ACE),SPOTT THUS
AC ** AT ** WITH FUEL ** SPOTT= **
      IF ((AC.LOCATION(ACE) < 9) OR (AC.FUEL.STAT(ACE) > .5))
        PRINT 1 LINE WITH ACE,SPOTT THUS
AC ** WILL ABORT MOVEMENT TO SPOT **
      LET FLAG = 1
      PRINT 1 LINE WITH EVENT.V,NUM.OPEN.SPOTS,SPOTT THUS
      EVENT ** #OPEN SPOTS ** OPEN SPOT * ELSE
      FOR EACH ACE IN THE SHIP,

```

```

        WITH AC.ID(ACE) = ABS.F(SPOT(SPOTT))
        FIND THE FIRST CASE
        PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),SPOTT THUS
        AC ** IN TRANSIT FROM ** TO ** MUST BE DISPLACED
        IF AC.LOCATION(ACE) = 9
            FOR EACH AC.RECOVERED IN EV.S(I.AC.RECOVERED)
                WITH AC2 = ACE, DO
                CANCEL AC.RECOVERED
            LOOP
        ELSE
            FOR EACH AC.RESPOTTED IN EV.S(I.AC.RESPOTTED)
                WITH AC3 = ACE, DO
                CANCEL AC.RESPOTTED
            LOOP
        ALWAYS
        FILE ACE IN SPOT.Q
    PRINT 1 LINE WITH ACE THUS
    AC ** FILED IN SPOT.Q
        ALWAYS
        ALWAYS
    ELSE
        FOR EACH ACE IN THE SHIP
            WITH AC.ID(ACE) = ABS.F(SPOT(SPOTT))
            FIND THE FIRST CASE
            IF FOUND
                IF (AC.DESTINATION(ACE) = 7) OR
                    (AC.DESTINATION(ACE) = 8)
                    LET FLAG = 2
                    PRINT 1 LINE WITH AC.ID(ACE) THUS
                    AC ** IS IN TRANSIT TO THE BONE...WILL TAKE THIS SPOT FOR THE EMERG
                    IF M.TUG.Q(ACE) = 1
                        ADD 1 TO TUG
                        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
                        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
                        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
                        SCHEDULE A BONE.ARRIVAL GIVING ACE
                            IN TTRESPOT MINUTES
                        REMOVE ACE FROM TUG.Q
                    PRINT 1 LINE WITH ACE THUS
                    AC ** REMOVED FROM TUG.Q
                ALWAYS
                ALWAYS
                ALWAYS
            LOOP
        IF FLAG = 0
            **ALL SPOTS CURRENTLY HAVE AN AC ON THEM..NEED TO LAUNCH OR RESPOT AC
            PRINT 1 LINE WITH AC.ID(AC) THUS
            MUST LAUNCH OR RESPOT AIRCRAFT TO ACCOMMODATE AC ** EMERGENCY
            LET MIN.LAUNCH.TIME = 9999
            LET MAX.LAUNCH.TIME = 0
            LET AC.MIN = 0
            LET AC.MAX = 0
            LET AC.ROY = 0
            FOR I BACK FROM 6 TO 1 BY 1, DO
                FOR EACH ACE IN THE SHIP
                    WITH AC.ID(ACE) = SPOT(I)
                    FIND THE FIRST CASE
                    IF FOUND
                        IF ((AC.LAUNCH.TIME(ACE) > MAX.LAUNCH.TIME)

```

```

        AND (AC.TYPE(ACE) <> 3))
        LET AC.MAX = ACE
        LET MAX.LAUNCH.TIME = AC.LAUNCH.TIME(ACE)
    ALWAYS
    IF ((AC.LAUNCH.TIME(ACE) < MIN.LAUNCH.TIME) AND
        (AC.TYPE(ACE) <> 3) AND
        (AC.OP.STAT(ACE) > .65) AND
        (AC.FUEL.STAT(ACE) > .5))
        LET AC.MIN = ACE
        LET MIN.LAUNCH.TIME = AC.LAUNCH.TIME(ACE)
    ALWAYS
    IF ((AC.FUEL.STAT(ACE)=1.)AND(AC.LOAD.STAT(ACE)= 1.)
        AND (AC.RDY = 0) AND (AC.TYPE(ACE) <> 3))
        LET AC.RDY = ACE
    ALWAYS
    PRINT 1 LINE WITH I,AC.ID(ACE),AC.MIN,AC.MAX THUS
I ** AC ** AC.MIN **** AC.MAX ****
    ELSE
        PRINT 1 LINE WITH I,SPOT(I),AC.MIN,AC.MAX THUS
I ** AC ** AC.MIN **** AC.MAX ****
    ALWAYS
    LOOP
    IF ((AC.RDY <> 0) AND (AC.LOAD.STAT(AC.MIN) < 1.))
        LET AC.MIN = AC.RDY
    ALWAYS
    IF (AC.MIN <> 0) AND (AC.TYPE(AC.MIN) < 3)
        LET SPOTT = AC.LOCATION(AC.MIN)
        PRINT 1 LINE WITH AC.ID(AC.MIN),SPOTT THUS
AC ** WILL LAUNCH IMMEDIATELY FROM SPOT * FOR THE EMERGENCY RECOVERY
        LET SPOT(SPOTT) = 0
        IF AC.LOAD.STAT(AC.MIN) = 1.0
            FOR EACH FLIGHTS IN PLAN
                FOR EACH ACE IN FLT.WAVE(FLIGHTS)
                    WITH ACE = AC.MIN
                FIND THE FIRST CASE
                IF FOUND
                    REMOVE AC.MIN FROM FLT.WAVE(FLIGHTS)
                    SUBTRACT 1 FROM FLT.AC.RDY(FLIGHTS)
                    SUBTRACT 1 FROM FLT.AC.NUM(FLIGHTS)
                    LET T.FLT.NUM = INT.F(FLT.NUM(FLIGHTS))
                    SCHEDULE AN AC.LAUNCHED GIVING AC.MIN,T.FLT.NUM
                        IN UNIFORM.F(.5,1.,5) MINUTES
                    PRINT 1 LINE WITH AC.ID(AC.MIN) THUS
AC ** SCHEDULED TO LAUNCH IN .5-1 MINUTES
                ELSE
                    PRINT 1 LINE THUS
**ERROR** FLIGHT NOT FOUND IN PLAN
                ALWAYS
            ELSE
                LET AC.FUEL.STAT(AC.MIN) = AC.FUEL.STAT(AC.MIN) - .02 **SINCE
                LET AC.PRIORITY(AC.MIN) = 1 - AC.FUEL.STAT(AC.MIN) * .2 **W
                LET AC.LOCATION(AC.MIN) = 10
                LET AC.DESTINATION(AC.MIN) = 9
                LET ETA(AC.ID(AC.MIN)) = 1 + TIME.V
                LET AC.OP.STAT(AC.MIN) = AC.OP.STAT(AC.MIN) + .2 **FOR LOAD
                SCHEDULE A DELTA.ARRIVAL GIVING AC.MIN **WHEN AC.
                    IN UNIFORM.F(1.,2.,5) MINUTES
            ALWAYS
            LET ACE = AC.MIN
        ELSE

```

```

      LET SPOTT = AC.LOCATION(AC.MAX)
      PRINT 1 LINE WITH AC.ID(AC.MAX), SPOTT THUS
AC ** WILL RESPOT IMMEDIATELY FROM SPOT * FOR THE EMERGENCY RECOVERY
      LET SPOT(SPOTT) = 0
      IF AC.TYPE(AC.MAX) = 1
        LET AC.DESTINATION(AC.MAX) = 7
        FILE AC.MAX IN BONE.FWD
      ELSE
        LET AC.DESTINATION(AC.MAX) = 8
        FILE AC.MAX IN BONE.AFT
      ALWAYS
      PRINT 1 LINE WITH N.BONE.FWD, N.BONE.AFT THUS
N.BONE.FWD = ** N.BONE.AFT = **
      LET XBAR = TTRESPOT.AC(AC.TYPE(AC.MAX))
      LET SOEV = S.TTRESPOT.AC(AC.TYPE(AC.MAX))
      LET TTRESPOT = NORMAL.F(XBAR, SOEV, 2)
      SCHEDULE A BONE.ARRIVAL GIVING AC.MAX IN TTRESPOT MINUTES
      LET ACE = AC.MAX
      ADD 1 TO TUG
      ALWAYS
      IF AC.SERVICE.FLAG(ACE) = 1
        IF AC.FUEL.STAT(ACE) < 1. **AVBS NEVER ENTER THIS SECTION
          FOR EACH AC.REFUELED IN EV.S(I.AC.REFUELED)
            WITH AC5 = ACE, DO
              CANCEL AC.REFUELED
              PRINT 1 LINE WITH ACE THUS
AC.REFUELED EVENT SCHEDULED FOR AC ** HAS BEEN CANCELLED
              SUBTRACT 1 FROM REFUELER
          LOOP
          ALWAYS
          IF AC.LOAD.STAT(ACE) < 1.
            FOR EACH AC.LOADED IN EV.S(I.AC.LOADED)
              WITH AC4 = ACE, DO
                CANCEL AC.LOADED
                PRINT 1 LINE WITH ACE THUS
AC.LOADED EVENT SCHEDULED FOR AC ** HAS BEEN CANCELLED
            LOOP
            ALWAYS
            LET AC.SERVICE.FLAG(ACE) = 0
          ALWAYS
          IF M.TUG.Q(ACE) = 1
            REMOVE THIS ACE FROM TUG.Q
            PRINT 1 LINE WITH ACE THUS
AC ** REMOVED FROM TUG.Q
          ALWAYS
          IF M.SPOT.Q(ACE) = 1
            REMOVE THIS ACE FROM SPOT.Q
          ALWAYS
          IF M.ELEVATOR.Q(ACE) = 1
            REMOVE THIS ACE FROM ELEVATOR.Q
          ALWAYS
          IF M.REFUELER.Q(ACE) = 1
            REMOVE THIS ACE FROM REFUELER.Q
          ALWAYS
      PRINT 1 LINE WITH AC.ID(AC), AC.ID(ACE) THUS
      AC = ** ACE = **

      IF (AC.FUEL.STAT(AC) < .25) AND (AC.TYPE(AC) = AC.TYPE(ACE)) **SW
        AND (AC.LOAD.STAT(ACE) <> 1.0)

```

```

IF AC.LAUNCH.TIME(ACE) < AC.LAUNCH.TIME(AC)
  FOR EACH FLIGHTE IN PLAN
    WITH FLT.TIME(FLIGHTE) = AC.LAUNCH.TIME(ACE)
  FIND THE FIRST CASE
  IF FOUND

    PRINT 1 LINE WITH FLT.TIME(FLIGHTE),
                      N.FLT.WAVE(FLIGHTE) THUS

  #AC IN FLIGHT SCHED FOR **** IS **
    FOR EACH ACE.P IN FLT.WAVE(FLIGHTE), DO
      PRINT 1 LINE WITH AC.ID(ACE.P), AC.LAUNCH.TIME(ACE.P) THUS
    AC ** IS SCHED TO LAUNCH AT ****
    LOOP

  PRINT 1 LINE WITH AC.ID(AC), AC.ID(ACE) THUS
  AC = **   ACE = **

  REMOVE ACE FROM FLT.WAVE(FLIGHTE)
  LET AC.LAUNCH.TIME(ACE) = AC.LAUNCH.TIME(AC)
  LET AC.LAUNCH.TIME(AC) = FLT.TIME(FLIGHTE)
  PRINT 1 LINE WITH AC.ID(AC), AC.ID(ACE) THUS
  AC = **   ACE = **

  PRINT 1 LINE WITH FLT.TIME(FLIGHTE),
                      N.FLT.WAVE(FLIGHTE) THUS

  #AC IN FLIGHT SCHED FOR **** IS **
    FOR EACH ACE.P IN FLT.WAVE(FLIGHTE), DO
      PRINT 1 LINE WITH AC.ID(ACE.P), AC.LAUNCH.TIME(ACE.P) THUS
    AC ** IS SCHED TO LAUNCH AT ****
    LOOP

  PRINT 1 LINE WITH AC.ID(AC), AC.ID(ACE) THUS
  AC = **   ACE = **

  ALWAYS
  IF AC.LAUNCH.TIME(ACE) < 9999
    FOR EACH FLIGHTE.P IN PLAN
      WITH FLT.TIME(FLIGHTE.P) = AC.LAUNCH.TIME(ACE)
    FIND THE FIRST CASE
    IF FOUND
      PRINT 1 LINE WITH AC.ID(AC), AC.ID(ACE) THUS
      AC = **   ACE = **

      PRINT 1 LINE WITH FLT.TIME(FLIGHTE.P),
                        N.FLT.WAVE(FLIGHTE.P) THUS

      #AC IN FLIGHT SCHED FOR **** IS **
        FOR EACH ACE.P IN FLT.WAVE(FLIGHTE.P), DO
          PRINT 1 LINE WITH AC.ID(ACE.P), AC.LAUNCH.TIME(ACE.P) THUS
        AC ** IS SCHED TO LAUNCH AT ****
        LOOP

      PRINT 1 LINE WITH AC.ID(AC), AC.ID(ACE) THUS
      AC = **   ACE = **

      REMOVE AC FROM FLT.WAVE(FLIGHTE.P)
      FILE ACE IN FLT.WAVE(FLIGHTE.P)
      PRINT 1 LINE WITH FLT.TIME(FLIGHTE.P),
                        N.FLT.WAVE(FLIGHTE.P) THUS

      #AC IN FLIGHT SCHED FOR **** IS **

```

```

PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **
      FOR EACH ACE.P IN FLT.WAVE(FLIGHTE.P), DO
      PRINT 1 LINE WITH AC.ID(ACE.P),AC.LAUNCH.TIME(ACE.P) THUS
AC ** IS SCHED TO LAUNCH AT ****
      LOOP

      ALWAYS
      ALWAYS
      FILE AC IN FLT.WAVE(FLIGHTE)
      PRINT 1 LINE WITH FLT.TIME(FLIGHTE),
      N.FLT.WAVE(FLIGHTE) THUS
OAC IN FLIGHT SCHED FOR **** IS **
      FOR EACH ACE.P IN FLT.WAVE(FLIGHTE), DO
      PRINT 1 LINE WITH AC.ID(ACE.P),AC.LAUNCH.TIME(ACE.P) THUS
AC ** IS SCHED TO LAUNCH AT ****
      LOOP

PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **
      ALWAYS
      ALWAYS
      ALWAYS
      LET AC.DESTINATION(AC) = SPOTT
      LET SPOT(SPOTT) = - AC.ID(AC)
      REMOVE THIS AC FROM DELTA.PATTERN
      REMOVE THIS AC FROM SPOT.Q

      LET XBAR = TTRECOVER.AC(AC.TYPE(AC))
      LET SDEV = S.TTRECOVER.AC(AC.TYPE(AC))
      LET TTRECOVER = NORMAL.P(XBAR,SDEV,1)
      IF TTRECOVER < (LAST.RECOVERY.TIME + .5 - TIME.V)
      LET TTRECOVER = LAST.RECOVERY.TIME + .5 - TIME.V
      ALWAYS
      IF TTRECOVER < (LAST.LAUNCH.TIME + 2 - TIME.V)
      LET TTRECOVER = LAST.LAUNCH.TIME + 2 - TIME.V
      ALWAYS
      LET LAST.RECOVERY.TIME = TTRECOVER + TIME.V

      SCHEDULE AN AC.RECOVERED GIVING AC IN TTRECOVER MINUTES
      ADD 1 TO NUM.EMERGENCY.RECOVERIES
      PRINT 1 LINE WITH AC.ID(AC),SPOTT,TTRECOVER THUS
AC ** WILL RECOVER TO SPOT ** IN **.* MINUTES
      FOR EACH ACE IN DELTA.PATTERN, DO
      PRINT 1 LINE WITH AC.ID(ACE),AC.PRIORITY(ACE),
      AC.DESTINATION(ACE) THUS
AC ** WITH PRIORITY **.* HAS DEST **
      LOOP
      FOR EACH ACE IN DELTA.PATTERN
      WITH ((AC.PRIORITY(ACE) > EMERGENCY.STAT.AC(AC.TYPE(ACE)))
      AND (AC.DESTINATION(ACE) = 9)),
      FIND THE FIRST CASE
      IF FOUND
      SCHEDULE A SPOT.EMERGENCY GIVING ACE IN .25 MINUTES
      PRINT 1 LINE WITH ACE,TIME.V THUS
AC ** DECLARES EMERGENCY AT **.*.
      ALWAYS
      FILE AC IN DELTA.PATTERN

```

ELSE
 PRINT 1 LINE WITH AC.ID(CAC) THUS
AC ** HAS ALREADY BEEN ASSIGNED A RECOVERY SPOT
ALWAYS
RETURN
END

```

** *****
EVENT SPOT.OPEN
** *****
DEFINE SPOT, FLAG, AC, ACE.T AS INTEGER VARIABLES
DEFINE INTERVAL, TTRECOVER AS REAL VARIABLES

** THIS SECTION OF CODE UPDATES THE FUEL AND PRIORITY STATUS VARIABLES OF
** THE AIRCRAFT IN THE DELTA PATTERN IF THE LAST UPDATE TIME WAS MORE THAN
** 10 MINUTES AGO.

LET J = 0
FOR I BACK FROM 6 TO 1 BY 1, DO
  IF SPOT(I) = 0
    ADD 1 TO J
  ALWAYS
LOOP
IF J <> NUM.OPEN.SPOTS
  PRINT 1 LINE WITH TIME.V, EVENT.V, J, NUM.OPEN.SPOTS THUS
**ERROR** TIME: ****. ** EVENT ** J = ** NUM.OPEN.SPOTS = **
  ALWAYS

LET FLAG = 0
PRINT 1 LINE WITH NUM.OPEN.SPOTS, DELTA.UPDATE.TIME,
  N.DELTA.PATTERN, N.SPOT.Q THUS
#OPEN SPOTS = * LAST DELTA UPDATE = **** #DELTA AC = ** N.SPOT.Q = **
IF N.DELTA.PATTERN > 0
  FOR EACH ACE IN DELTA.PATTERN, DO
    LET DELTA = ((TIME.V - DELTA.UPDATE.TIME)/60) *
      (FUELUSE.AC(AC.TYPE(ACE)))
    LET AC.FUEL.STAT(ACE) = AC.FUEL.STAT(ACE) - DELTA
    LET AC.FLYING.TIME(ACE) = (AC.FUEL.STAT(ACE)
      / FUELUSE.AC(AC.TYPE(ACE))) * 60.
    LET AC.PRIORITY(ACE) = AC.PRIORITY(ACE) + DELTA * .2
  PRINT 1 LINE WITH AC.ID(ACE), AC.DESTINATION(ACE), AC.PRIORITY(ACE),
    AC.FUEL.STAT(ACE), AC.FLYING.TIME(ACE) THUS
  AC ** WITH DEST ** HAS PRIORITY *. ** FUEL.STAT *. ** FLYING.TIME **. **
  REMOVE THIS ACE FROM THE DELTA.PATTERN
  FILE THIS ACE IN THE DELTA.PATTERN
LOOP
LET DELTA.UPDATE.TIME = TIME.V
FOR EACH ACE IN DELTA.PATTERN,
  WITH (AC.PRIORITY(ACE) > EMERGENCY.STAT.AC(AC.TYPE(ACE)))
  AND (AC.DESTINATION(ACE) = 9)
  FIND THE FIRST CASE
  IF FOUND
    SCHEDULE A SPOT.EMERGENCY GIVING ACE IN .25 MINUTES
    PRINT 1 LINE WITH AC.ID(ACE), TIME.V THUS
  AC ** DECLARES EMERGENCY AT ****. **
  LET FLAG = 1
ELSE
  FOR EACH ACE IN DELTA.PATTERN,
    WITH (AC.PRIORITY(ACE) > PRIORITY.STAT.AC(AC.TYPE(ACE)))
    AND (AC.DESTINATION(ACE) = 9)
    FIND THE FIRST CASE
    IF FOUND
      SCHEDULE A SPOT.PRIORITY GIVING ACE IN 1 MINUTE
      PRINT 1 LINE WITH AC.ID(ACE), TIME.V THUS
  AC ** DECLARES PRIORITY AT ****. **
  LET FLAG = 1

```

```

        ALWAYS
    ALWAYS
ALWAYS
IF FLAG = 0
    LET INTERVAL = 0
    LET I = 7

    FOR EACH ACE IN SPOT.Q, DO
        IF AC.LOCATION(ACE) = 9
            LET INDEX(AC.ID(ACE)) = 8
        ELSE
            LET INDEX(AC.ID(ACE)) = 7
        ALWAYS
        PRINT 1 LINE WITH AC.ID(ACE), AC.LOCATION(ACE),
            AC.LAUNCH.TIME(ACE) THUS
AC ** AT * SCHED TO LAUNCH AT ***
    LOOP

    WHILE ((NUM.OPEN.SPOTS > 0) AND (N.SPOT.Q > 0) AND (I > 1)), DO
        SUBTRACT 1 FROM I

        PRINT 1 LINE WITH NUM.OPEN.SPOTS, N.SPOT.Q, I, SPOT(I) THUS
        #OPEN SPOTS= ** #SPOT.Q = ** I = * SPOT(I) = **

        IF SPOT(I) = 0
            LET SPOTT = I

            IF N.AV8.PLAN > 0
                REMOVE THE FIRST FLIGHTS FROM AV8.PLAN
                LET AV8.LAUNCH.TIME = FLT.TIME(FLIGHTS)
                FILE THIS FLIGHTS IN AV8.PLAN
            ELSE
                LET AV8.LAUNCH.TIME = 9999
            ALWAYS

            FOR EACH FLIGHTS IN THE PLAN
                WITH ((FLT.AC.TYPE(FLIGHTS) <> 3)
                    AND (FLT.DELAY(FLIGHTS) = 3))
                FIND THE FIRST CASE
                IF FOUND
                    LET HELD.LAUNCH.TIME = FLT.TIME(FLIGHTS)
                ELSE
                    LET HELD.LAUNCH.TIME = 9999
                ALWAYS

            PRINT 1 LINE WITH AV8.LAUNCH.TIME, HELD.LAUNCH.TIME,
                SPOTT THUS
AV8.LAUNCH.TIME= *** HELD.LAUNCH.TIME= *** OPEN SPOT IS **

        FOR EACH ACE IN SPOT.Q,
            FOR J = 1 TO SPOT.AC(AC.TYPE(ACE), INDEX(AC.ID(ACE))),
                WITH ((SPOT.AC(AC.TYPE(ACE), J) = SPOTT)
                    AND ((AC.LOCATION(ACE) = 9)
                        AND ((AC.FLYING.TIME(ACE) < 33)
                            OR (((AC.LAUNCH.TIME(ACE) - TIME.V)
                                > AC.FLYING.TIME(ACE))
                                OR ((AC.LAUNCH.TIME(ACE) - TIME.V) < 25))
                            AND ((AV8.LAUNCH.TIME

```

```

        > AC.LAUNCH.TIME(ACE))
        OR (AV8.LAUNCH.TIME > (TIME.V + 20)))
    OR ((AC.TYPE(ACE) = 3)
        AND (AV8.LAUNCH.TIME > (TIME.V + 10)))
    AND ((HELD.LAUNCH.TIME > (TIME.V + 10))
        OR ((HELD.LAUNCH.TIME + 1.)
            > AC.LAUNCH.TIME(ACE))))))
    OR ((AC.LOCATION(ACE) < 9)
        AND ((SPOTT > 2)
            OR (AV8.LAUNCH.TIME >= AC.LAUNCH.TIME(ACE)))
        AND ((AC.LAUNCH.TIME(ACE) - TIME.V) < 25))))
    FIND THE FIRST CASE
    IF FOUND ** AC COMPATIBLE WITH SPOT IDENTIFIED

    PRINT 1 LINE WITH AC.ID(ACE),SPOTT,AC.LAUNCH.TIME(ACE),
        AC.FLYING.TIME(ACE)  THUS
    AC ** COMPATIBLE TO GO TO SPOT * HAS LAUNCH.TIME= *** AND FLYING.TIME= *

```

```

IF AC.LOCATION(ACE) = 9

```

```

    IF FLAG = 0
    LET XBAR = TTRECOVER.AC(AC.TYPE(ACE))
    LET SDEV = S.TTRECOVER.AC(AC.TYPE(ACE))
    LET TTRECOVER = NORMAL.F(XBAR,SDEV,1)
    IF TTRECOVER < (LAST.RECOVERY.TIME + .5 - TIME.V)
        LET TTRECOVER = LAST.RECOVERY.TIME + .5 - TIME.V
    ALWAYS
    IF TTRECOVER < (LAST.LAUNCH.TIME + 2 - TIME.V)
        LET TTRECOVER = LAST.LAUNCH.TIME + 2 - TIME.V
    ALWAYS
    LET FLAG = 1
    ALWAYS

```

```

    LET AC.DESTINATION(ACE) = SPOTT
    LET SPOT(SPOTT) = -AC.ID(ACE)
    SUBTRACT 1 FROM NUM.OPEN.SPOTS
    PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
    NUM.OPEN.SPOTS = **
    SCHEDULE AN AC.RECOVERED GIVING ACE
        IN TTRECOVER + INTERVAL MINUTES
    ADD UNIFORM.F(.5,.8,1) TO INTERVAL
    LET LAST.RECOVERY.TIME = TTRECOVER + TIME.V
        + INTERVAL
    PRINT 1 LINE WITH AC.ID(ACE),SPOTT,
        (TTRECOVER+INTERVAL)  THUS
    AC ** WILL RECOVER TO SPOT ** IN ***. MINUTES
    REMOVE THIS ACE FROM SPOT.Q
    ELSE
    IF AC.LOCATION(ACE) < 12
    IF AC.TYPE(ACE) < 3
        LET AC.DESTINATION(ACE) = SPOTT

```

```

    LET SPOT(SPOTT) = -AC.ID(ACE)
    IF AC.SERVICE.FLAG(ACE) = 0
      IF TUG < 4
        ADD 1 TO TUG
        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE AN AC.RESPOTTED GIVING ACE
          IN TTRESPOT MINUTES
        PRINT 1 LINE WITH AC.ID(ACE),SPOTT THUS
        AC ** WILL RESPOT TO SPOT **
      ELSE
        FILE THIS ACE IN TUG.Q
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN TUG.Q
    ALWAYS
    ALWAYS
    IF AC.LOCATION(ACE) > 6
      SUBTRACT 1 FROM NUM.OPEN.SPOTS
      PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
      NUM.OPEN.SPOTS = **
    ALWAYS
    REMOVE THIS ACE FROM SPOT.Q
  ELSE
    IF AC.LAUNCH.TIME(ACE) = AV8.LAUNCH.TIME
      IF SPOTT <= 2 **AV8'S CAN RECOVER ON 1,2,5,6 BUT L
      IF (AC.LAUNCH.TIME(ACE) - TIME.V) <= 10.
        LET AC.DESTINATION(ACE) = SPOTT
        LET SPOT(SPOTT) = -AC.ID(ACE)
        LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
        LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
        LET TTRESPOT = NORMAL.F(XBAR,SDEV,2)
        SCHEDULE AN AC.RESPOTTED GIVING ACE IN
          TTRESPOT MINUTES
        PRINT 1 LINE WITH AC.ID(ACE),SPOTT THUS
        AC = ** WILL RESPOT TO **
        SUBTRACT 1 FROM NUM.OPEN.SPOTS
        PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
        NUM.OPEN.SPOTS = **
        REMOVE THIS ACE FROM SPOT.Q
      ELSE
        SCHEDULE A SPOT.OPEN IN (AC.LAUNCH.TIME(ACE)
          - 10. + UNIFORM.F(.1,2,.9)) MINUTES
    ALWAYS
    ALWAYS
    ALWAYS
    ALWAYS
  ELSE
    IF AC.TYPE(ACE) < 3
      LET AC.DESTINATION(ACE) = SPOTT
      LET SPOT(SPOTT) = - AC.ID(ACE)
      SUBTRACT 1 FROM NUM.OPEN.SPOTS
      PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
      NUM.OPEN.SPOTS = **
    ELSE
      LET AC.DESTINATION(ACE) = 8
      FILE ACE IN BONE.AFT
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN BONE
    ALWAYS

```

```

      IF (SPOT(11) = 0) AND
        (CHANGER.EQUIV < MAX.HANGER.EQUIV)
        LET XBAR = TTARRIV.E.AC(AC.TYPE(ACE))
        LET SDEV = S.TTARRIV.E.AC(AC.TYPE(ACE))
        LET TTARRIV.E = NORMAL.FC(XBAR,SDEV,8)
        SCHEDULE AN ELEVATOR.ARRIVAL GIVING ACE
          IN TTARRIV.E MINUTES
        LET SPOT(11) = -AC.ID(ACE)
      ELSE
        IF M.ELEVATOR.Q(ACE) <> 1
          FILE ACE IN ELEVATOR.Q
          ALWAYS
        PRINT 1 LINE WITH ACE THUS
        AC ** FILED IN ELEVATOR.Q
          ALWAYS
          REMOVE ACE FROM SPOT.Q
          ALWAYS
          ALWAYS
          PRINT 1 LINE WITH N.SPOT.Q,NUM.OPEN.SPOTS,I,SPOT(I),
            AC.ID(ACE), AC.LOCATION(ACE) THUS
        *IN SPOT.Q ** *OPEN SPOTS * I * SPOT(I) ** AC.ID ** AC.LOC **
          ALWAYS
          ALWAYS
          LOOP
          ALWAYS
          RETURN
          END

```

```

*****
EVENT SPOT.PRIORITY GIVEN AC
*****
DEFINE SPOTT, AC, ACE, ACE.T, FLAG AS INTEGER VARIABLES
DEFINE INTERVAL AS A REAL VARIABLE
PRINT 1 LINE WITH TIME.V, EVENT.V, AC.ID(AC), AC.PRIORITY(AC),
AC.FUEL.STAT(AC) THUS
T ****. ** E ** AC ** W PRI *.*** FUEL *.** REQUESTED PRIORITY LANDING

IF M.SPOT.Q(AC) = 1 **OTHERWISE AC ALREADY HAS A RECOVERY DESTINATION
  IF NUM.OPEN.SPOTS > 0
    FOR I BACK FROM 6 TO 1 BY 1
      WITH SPOT(I) = 0
      FIND THE FIRST CASE
      IF FOUND
        LET SPOTT = I
      ELSE
        LET SPOTT = 0
        PRINT 1 LINE THUS
**ERROR** OPEN SPOT NOT FOUND
      ALWAYS
    ALWAYS
  IF I <> 0

```

```

PRINT 1 LINE WITH EVENT.V, NUM.OPEN.SPOTS, SPOTT THUS
EVENT ** #OPEN SPOTS ** OPEN SPOT * IF
  LET AC.DESTINATION(AC) = SPOTT
  LET SPOT(SPOTT) = - AC.ID(AC)
  SUBTRACT 1 FROM NUM.OPEN.SPOTS
  PRINT 1 LINE WITH NUM.OPEN.SPOTS THUS
  NUM.OPEN.SPOTS = **
  REMOVE THIS AC FROM DELTA.PATTERN
  REMOVE THIS AC FROM SPOT.Q

```

```

LET XBAR = TTRECOVER.AC(AC.TYPE(AC))
LET SDEV = S.TTRECOVER.AC(AC.TYPE(AC))
LET TTRECOVER = NORMAL.F(XBAR, SDEV, 1)
IF TTRECOVER < (LAST.RECOVERY.TIME + .5 - TIME.V)
  LET TTRECOVER = LAST.RECOVERY.TIME + .5 - TIME.V
ALWAYS
IF TTRECOVER < (LAST.LAUNCH.TIME + 2 - TIME.V)
  LET TTRECOVER = LAST.LAUNCH.TIME + 2 - TIME.V
ALWAYS
LET LAST.RECOVERY.TIME = TTRECOVER + TIME.V

```

```

SCHEDULE AN AC.RECOVERED GIVING AC IN TTRECOVER MINUTES
ADD 1 TO NUM.PRIORITY.RECOVERIES
PRINT 1 LINE WITH EVENT.V, AC.ID(AC), SPOTT THUS

```

```

E = ** AC ** WILL RECOVER TO SPOT **
      FOR EACH ACE IN DELTA.PATTERN, DO
        PRINT 1 LINE WITH
          AC.ID(ACE),AC.PRIORITY(ACE),AC.DESTINATION(ACE),
          AC.FLYING.TIME(ACE) THUS
AC ** WITH PRIORITY **.*** HAS DEST ** AND FLYING.TIME **.*
      LOOP
      FOR EACH ACE IN DELTA.PATTERN
        WITH ((AC.PRIORITY(ACE) > EMERGENCY.STAT.AC(AC.TYPE(ACE)))
          AND (AC.DESTINATION(ACE) = 9)),
        FIND THE FIRST CASE
        IF FOUND
          SCHEDULE A SPOT.EMERGENCY GIVING ACE IN .25 MINUTES
          PRINT 1 LINE WITH ACE,TIME.V THUS
AC ** DECLARES EMERGENCY AT **.***
      ELSE
        FOR EACH ACE IN DELTA.PATTERN
          WITH ((AC.PRIORITY(ACE) > PRIORITY.STAT.AC(AC.TYPE(ACE)))
            AND (AC.DESTINATION(ACE) = 9)),
          FIND THE FIRST CASE
          IF FOUND
            SCHEDULE A SPOT.PRIORITY GIVING ACE IN 1 MINUTES
            PRINT 1 LINE WITH ACE,TIME.V THUS
AC ** DECLARES PRIORITY AT **.***
      ALWAYS
      ALWAYS
      FILE AC IN DELTA.PATTERN
    ELSE
      PRINT 1 LINE THUS
NO OPEN SPOTS
      LET FLAG = 0
      FOR I BACK FROM 6 TO 1 BY 1
        WHILE FLAG = 0, DO
          LET SPOTT = I
          IF SPOT(SPOTT) < 0
            FOR EACH ACE IN THE SHIP
              WITH AC.ID(ACE) = - SPOT(SPOTT)
              FIND THE FIRST CASE
              IF FOUND
                PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),
                  AC.PRIORITY(ACE), AC.DESTINATION(ACE),
                  AC.FUEL.STAT(ACE), AC.LAUNCH.TIME(ACE) THUS
AC ** AT ** HAS PRIORITY **.*** DEST ** FUEL **.*** AND LAUNCH TIME **.*
                IF ((AC.LOCATION(ACE) < 9) OR ((AC.FUEL.STAT(ACE) > .5)
                  AND (AC.LAUNCH.TIME(ACE) > (TIME.V+15))
                  AND (AC.PRIORITY(ACE) < .90)
                  AND (AC.TYPE(ACE) <> 3)))
                  LET FLAG = 1
                PRINT 1 LINE WITH EVENT.V,NUM.OPEN.SPOTS,SPOTT THUS
                EVENT ** #OPEN SPOTS ** TARGET SPOT * ELSE

                PRINT 1 LINE WITH AC.ID(ACE),AC.LOCATION(ACE),SPOTT THUS
                AC ** IN TRANSIT FROM ** TO ** MUST BE DISPLACED
                IF AC.LOCATION(ACE) = 9
                  FOR EACH AC.RECOVERED IN EV.S(I.AC.RECOVERED)
                    WITH AC2 = ACE, DO
                      CANCEL AC.RECOVERED
                      PRINT 1 LINE WITH AC.ID(ACE) THUS
AC.RECOVERED EVENT SCHEDULED FOR AC ** HAS BEEN CANCELLED
                LOOP

```

```

ELSE
  FOR EACH AC.RESPOTTED IN EV.S(I.AC.RESPOTTED)
    WITH AC3 = ACE, DO
      CANCEL AC.RESPOTTED
      PRINT 1 LINE WITH AC.ID(ACE) THUS
AC.RESPOTTED EVENT SCHEDULED FOR AC ** HAS BEEN CANCELLED
  LOOP
  ALWAYS
  FILE ACE IN SPOT.Q
  LET AC.DESTINATION(ACE) = AC.LOCATION(ACE)
PRINT 1 LINE WITH ACE THUS
AC ** FILED IN SPOT.Q
  ALWAYS
  ELSE
    PRINT 1 LINE THUS
**ERROR** AC NOT FOUND
  ALWAYS
  ELSE
    FOR EACH ACE IN THE SHIP
      WITH AC.ID(ACE) = SPOT(SPOTT)
      FIND THE FIRST CASE
      IF FOUND
        PRINT 1 LINE WITH AC.ID(ACE), AC.LOCATION(ACE),
          AC.FUEL.STAT(ACE), AC.LAUNCH.TIME(ACE) THUS
AC ** AT ** HAS FUEL *** AND LAUNCH TIME ***
        IF (AC.DESTINATION(ACE) = 7) OR
          (AC.DESTINATION(ACE) = 8)
          LET FLAG = 2
          PRINT 1 LINE WITH AC.ID(ACE) THUS
AC ** IS IN TRANSIT TO THE BONE...WILL TAKE THIS SPOT FOR THE PRIORITY
          IF M.TUG.Q(ACE) = 1
            ADD 1 TO TUG
            LET XBAR = TTRESPOT.AC(AC.TYPE(ACE))
            LET SDEV = S.TTRESPOT.AC(AC.TYPE(ACE))
            LET TTRESPOT = NORMAL.F(XBAR, SDEV, 2)
            SCHEDULE A BONE.ARRIVAL GIVING ACE
              IN TTRESPOT MINUTES
            REMOVS ACE FROM TUG.Q
          PRINT 1 LINE WITH ACE THUS
AC ** REMOVED FROM TUG.Q
          ALWAYS
          ALWAYS
          ALWAYS
          ALWAYS
        LOOP
        IF FLAG <> 0
          **
          IF (AC.FUEL.STAT(AC) < .25) AND (AC.TYPE(AC) = AC.TYPE(ACE)) **SWI
            AND (AC.LOAD.STAT(ACE) <> 1.0)
          IF AC.LAUNCH.TIME(ACE) < AC.LAUNCH.TIME(AC)
            FOR EACH FLIGHT IN PLAN
              WITH FLT.TIME(FLIGHT) = AC.LAUNCH.TIME(ACE)
              FIND THE FIRST CASE
              IF FOUND
                PRINT 1 LINE WITH FLT.TIME(FLIGHT),
                  N.FLT.WAVE(FLIGHT) THUS
**AC IN FLIGHT SCHED FOR *** IS **
                FOR EACH ACE.P IN FLT.WAVE(FLIGHT), DO

```

```

        PRINT 1 LINE WITH AC.ID(ACE.P),AC.LAUNCH.TIME(ACE.P) THUS
AC ** IS SCHED TO LAUNCH AT ****
    LOOP

PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **

        REMOVE ACE FROM FLT.WAVE(FLIGHTE)
        LET AC.LAUNCH.TIME(ACE) = AC.LAUNCH.TIME(AC)
        LET AC.LAUNCH.TIME(AC) = FLT.TIME(FLIGHTE)
PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **

        PRINT 1 LINE WITH FLT.TIME(FLIGHTE),
        N.FLT.WAVE(FLIGHTE) THUS
*AC IN FLIGHT SCHED FOR **** IS **
    FOR EACH ACE.P IN FLT.WAVE(FLIGHTE), DO
        PRINT 1 LINE WITH AC.ID(ACE.P),AC.LAUNCH.TIME(ACE.P) THUS
AC ** IS SCHED TO LAUNCH AT ****
    LOOP
PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **

        ALWAYS
        IF AC.LAUNCH.TIME(ACE) < 9999
            FOR EACH FLIGHTE.P IN PLAN
                WITH FLT.TIME(FLIGHTE.P) = AC.LAUNCH.TIME(ACE)
                FIND THE FIRST CASE
                IF FOUND
PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **
            PRINT 1 LINE WITH FLT.TIME(FLIGHTE.P),
            N.FLT.WAVE(FLIGHTE.P) THUS
*AC IN FLIGHT SCHED FOR **** IS **
    FOR EACH ACE.P IN FLT.WAVE(FLIGHTE.P), DO
        PRINT 1 LINE WITH AC.ID(ACE.P),AC.LAUNCH.TIME(ACE.P) THUS
AC ** IS SCHED TO LAUNCH AT ****
    LOOP

PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **
        REMOVE AC FROM FLT.WAVE(FLIGHTE.P)
        FILE ACE IN FLT.WAVE(FLIGHTE.P)
        PRINT 1 LINE WITH FLT.TIME(FLIGHTE.P),
        N.FLT.WAVE(FLIGHTE.P) THUS
*AC IN FLIGHT SCHED FOR **** IS **
PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **    ACE = **
    FOR EACH ACE.P IN FLT.WAVE(FLIGHTE.P), DO
        PRINT 1 LINE WITH AC.ID(ACE.P),AC.LAUNCH.TIME(ACE.P) THUS
AC ** IS SCHED TO LAUNCH AT ****
    LOOP

        ALWAYS
        ALWAYS
        FILE AC IN FLT.WAVE(FLIGHTE)
        PRINT 1 LINE WITH FLT.TIME(FLIGHTE),
        N.FLT.WAVE(FLIGHTE) THUS
*AC IN FLIGHT SCHED FOR **** IS **
    FOR EACH ACE.P IN FLT.WAVE(FLIGHTE), DO

```

```

        PRINT 1 LINE WITH AC.ID(ACE.P),AC.LAUNCH.TIME(ACE.P) THUS
AC ** IS SCHED TO LAUNCH AT ****
        LOOP

```

```

PRINT 1 LINE WITH AC.ID(AC),AC.ID(ACE) THUS
AC = **   ACE = **
          ALWAYS
          ALWAYS

```

```

..

```

```

LET AC.DESTINATION(AC) = SPOTT
LET SPOT(SPOTT) = - AC.ID(AC)
REMOVE THIS AC FROM DELTA.PATTERN
REMOVE THIS AC FROM SPOT.Q

```

```

LET XBAR = TTRECOVER.AC(AC.TYPE(AC))
LET SOEV = S.TTRECOVER.AC(AC.TYPE(AC))
LET TTRECOVER = NORMAL.F(XBAR,SOEV,1)
IF TTRECOVER < (LAST.RECOVERY.TIME + .5 - TIME.V)
    LET TTRECOVER = LAST.RECOVERY.TIME + .5 - TIME.V
ALWAYS
IF TTRECOVER < (LAST.LAUNCH.TIME + 2 - TIME.V)
    LET TTRECOVER = LAST.LAUNCH.TIME + 2 - TIME.V
ALWAYS
LET LAST.RECOVERY.TIME = TTRECOVER + TIME.V

```

```

SCHEDULE AN AC.RECOVERED GIVING AC IN TTRECOVER MINUTES
ADD 1 TO NUM.PRIORITY.RECOVERIES
PRINT 1 LINE WITH EVENT.V,AC.ID(AC),SPOTT,TTRECOVER THUS
E = **   AC ** WILL RECOVER TO SPOT ** IN ****.* MINUTES
FILE AC IN DELTA.PATTERN
FOR EACH ACE IN DELTA.PATTERN
    WITH ((AC.PRIORITY(ACE) > EMERGENCY.STAT.AC(AC.TYPE(ACE)))
        AND (AC.DESTINATION(ACE) = 9)),
    FIND THE FIRST CASE
    IF FOUND
        SCHEDULE A SPOT.EMERGENCY GIVING ACE IN .25 MINUTES
        PRINT 1 LINE WITH ACE,TIME.V THUS
AC **   DECLARES EMERGENCY AT ****.**
    ELSE
        FOR EACH ACE IN DELTA.PATTERN
            WITH ((AC.PRIORITY(ACE) > PRIORITY.STAT.AC(AC.TYPE(ACE)))
                AND (AC.DESTINATION(ACE) = 9)),
            FIND THE FIRST CASE
            IF FOUND
                SCHEDULE A SPOT.PRIORITY GIVING ACE IN 1 MINUTES
                PRINT 1 LINE WITH ACE,TIME.V THUS
AC **   DECLARES PRIORITY AT ****.**
    ALWAYS
    ALWAYS
    ALWAYS
ELSE
    PRINT 1 LINE WITH AC.ID(AC) THUS
AC ** HAS ALREADY BEEN ASSIGNED A RECOVERY SPOT
ALWAYS
FOR EACH ACE IN DELTA.PATTERN, DO
    PRINT 1 LINE WITH AC.ID(ACE),AC.PRIORITY(ACE),
        AC.DESTINATION(ACE) THUS

```

```

AC ** WITH PRIORITY **.*** HAS DEST **
  LOOP
  FOR EACH ACE IN DELTA.PATTERN
    WITH ((AC.PRIORITY(ACE) > EMERGENCY.STAT.AC(AC.TYPE(ACE)))
      AND (AC.DESTINATION(ACE) = 9)),
    FIND THE FIRST CASE
    IF FOUND
      SCHEDULE A SPOT.EMERGENCY GIVING ACE IN .25 MINUTES
      PRINT 1 LINE WITH ACE,TIME.V THUS
AC ** DECLARES EMERGENCY AT **.***
  ELSE
    FOR EACH ACE IN DELTA.PATTERN
      WITH ((AC.PRIORITY(ACE) > PRIORITY.STAT.AC(AC.TYPE(ACE)))
        AND (AC.DESTINATION(ACE) = 9)),
      FIND THE FIRST CASE
      IF FOUND
        SCHEDULE A SPOT.PRIORITY GIVING ACE IN 1 MINUTES
        PRINT 1 LINE WITH ACE,TIME.V THUS
AC ** DECLARES PRIORITY AT **.***
  ALWAYS
  ALWAYS
  RETURN
  END

```

ROUTINE TRACE
PRINT 3 LINES THUS

```
IF EVENT.V = 1
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC1(AC.LAUNCHED)),
    AC.LOCATION(AC1(AC.LAUNCHED)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 2
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC2(AC.RECOVERED)),
    AC.DESTINATION(AC2(AC.RECOVERED)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.DEST: **
ALWAYS
IF EVENT.V = 3
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC3(AC.RESPOTTED)),
    AC.DESTINATION(AC3(AC.RESPOTTED)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.DEST: **
ALWAYS
IF EVENT.V = 4
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC4(AC.LOADED)),
    AC.LOCATION(AC4(AC.LOADED)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 5
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC5(AC.REFUELED)),
    AC.LOCATION(AC5(AC.REFUELED)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 6
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC6(3ONE.ARRIVAL)),
    AC.LOCATION(AC6(3ONE.ARRIVAL)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 7
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC7(DELTA.ARRIVAL)),
    AC.LOCATION(AC7(DELTA.ARRIVAL)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 12
  PRINT 1 LINE WITH TIME.V,EVENT.V THUS
    TIME: ****.** EVENT: **
ALWAYS
IF EVENT.V = 13
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC8(SPOT.EMERGENCY)),
    AC.LOCATION(AC8(SPOT.EMERGENCY)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 14
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC9(SPOT.PRIGRITY)),
    AC.LOCATION(AC9(SPOT.PRIGRITY)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 15
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC10(MANGER.ARRIVAL)),
    AC.LOCATION(AC10(MANGER.ARRIVAL)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
```

```

IF EVENT.V = 16
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC11(ELEVATOR.ARRIVAL)),
    AC.LOCATION(AC11(ELEVATOR.ARRIVAL)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 17
  PRINT 1 LINE WITH TIME.V,EVENT.V,AC.ID(AC12(DECK.ARRIVAL)),
    AC.LOCATION(AC12(DECK.ARRIVAL)) THUS
    TIME: ****.** EVENT: ** AC.ID: ** AC.LOC: **
ALWAYS
IF EVENT.V = 8
  PRINT 1 LINE WITH TIME.V,EVENT.V,FLT.TIME(FLIGHT1(DECK.DECISION)),
    FLT.AC.TYPE(FLIGHT1(DECK.DECISION)) THUS
    TIME: ****.** EVENT: ** FLT.TIME **** FLT.TYPE **
ALWAYS
IF EVENT.V = 9
  PRINT 1 LINE WITH TIME.V,EVENT.V THUS
    TIME: ****.** EVENT: **
ALWAYS
IF EVENT.V = 10
  PRINT 1 LINE WITH TIME.V,EVENT.V,FLT.TIME(FLIGHT2(FLIGHT.LAUNCH)),
    FLT.AC.TYPE(FLIGHT2(FLIGHT.LAUNCH)) THUS
    TIME: ****.** EVENT: ** FLT.TIME **** FLT.TYPE **
ALWAYS
IF EVENT.V = 18
  PRINT 1 LINE WITH TIME.V,EVENT.V,FLT.TIME(FLIGHT3(FLIGHT.CHECK)),
    FLT.AC.TYPE(FLIGHT3(FLIGHT.CHECK)) THUS
    TIME: ****.** EVENT: ** FLT.TIME **** FLT.TYPE **
ALWAYS
RETURN
END

```

APPENDIX B

EVENT NUMBER KEY FOR SAMPLE VERIFICATION OUTPUT:

1	AC.LAUNCHED
2	AC.RECOVERED
3	AC.RESPOTTED
4	AC.LOADED
5	AC.REFUELED
6	BONE.ARRIVAL
7	DELTA.ARRIVAL
8	DECK.DECISION
9	SPOT.OPEN
10	FLIGHT.LAUNCH
11	STOP.SIMULATION
12	DELTA.UPDATE.CHK
13	SPOT.EMERGENCY
14	SPOT.PRIORITY
15	HANGER.ARRIVAL
16	ELEVATOR.ARRIVAL
17	DECK.ARRIVAL
18	FLIGHT.CHECK

8
 SEED 1 = 889656876
 SEED 2 = 575720521
 SEED 3 = 251431696
 SEED 4 = 1230229622
 SEED 5 = 1820681923
 SEED 6 = 481856962
 SEED 7 = 155710971
 SEED 8 = 1492838383
 SEED 9 = 1313992727

HANGER IS CAPABLE OF SLASHING 14 CH-46 EQUIVALENTS
 DISTANCE TO SHORE: 80000 METERS
 TYPES.AC = 3

NUM AC : 22

ID: 1	LQC: 7	FUEL: 1.00	LOAD: 0.	OP: 0.90	TYPE: 1
ID: 2	LQC: 7	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 1
ID: 3	LQC: 7	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 1
ID: 4	LQC: 7	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 1
ID: 5	LQC: 9	FUEL: 1.00	LOAD: 0.	OP: 1.00	TYPE: 1
ID: 6	LQC: 9	FUEL: 1.00	LOAD: 0.	OP: 1.00	TYPE: 1
ID: 7	LQC: 9	FUEL: 1.00	LOAD: 0.	OP: 1.00	TYPE: 1
ID: 8	LQC: 9	FUEL: 1.00	LOAD: 0.	OP: 1.00	TYPE: 1
ID: 9	LQC: 7	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 1
ID: 10	LQC: 7	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 1
ID: 11	LQC: 8	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 2
ID: 12	LQC: 8	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 2
ID: 13	LQC: 9	FUEL: 1.00	LOAD: 0.	OP: 1.00	TYPE: 2
ID: 14	LQC: 9	FUEL: 1.00	LOAD: 0.	OP: 1.00	TYPE: 2
ID: 15	LQC: 8	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 2
ID: 16	LQC: 8	FUEL: 1.00	LOAD: 0.	OP: 0.80	TYPE: 2
ID: 17	LQC: 8	FUEL: 1.00	LOAD: 1.00	OP: 1.00	TYPE: 3
ID: 18	LQC: 8	FUEL: 1.00	LOAD: 1.00	OP: 1.00	TYPE: 3
ID: 19	LQC: 8	FUEL: 1.00	LOAD: 1.00	OP: 1.00	TYPE: 3
ID: 20	LQC: 8	FUEL: 1.00	LOAD: 1.00	OP: 1.00	TYPE: 3
ID: 21	LQC: 12	FUEL: 0.	LOAD: 0.	OP: 0.60	TYPE: 3
ID: 22	LQC: 12	FUEL: 0.	LOAD: 0.	OP: 0.60	TYPE: 3

* SCHEDULED FLIGHTS = 35

I : 1	FLIGHT.TIME : 20	AC.TYPE : 1	#AC : 4
I : 2	FLIGHT.TIME : 21	AC.TYPE : 2	#AC : 2
I : 3	FLIGHT.TIME : 35	AC.TYPE : 1	#AC : 4
I : 4	FLIGHT.TIME : 36	AC.TYPE : 2	#AC : 2
I : 5	FLIGHT.TIME : 45	AC.TYPE : 3	#AC : 2
I : 6	FLIGHT.TIME : 65	AC.TYPE : 1	#AC : 2
I : 7	FLIGHT.TIME : 66	AC.TYPE : 2	#AC : 2
I : 8	FLIGHT.TIME : 75	AC.TYPE : 3	#AC : 2
I : 9	FLIGHT.TIME : 95	AC.TYPE : 1	#AC : 4
I : 10	FLIGHT.TIME : 96	AC.TYPE : 2	#AC : 2
I : 11	FLIGHT.TIME : 105	AC.TYPE : 3	#AC : 2
I : 12	FLIGHT.TIME : 125	AC.TYPE : 1	#AC : 4
I : 13	FLIGHT.TIME : 126	AC.TYPE : 2	#AC : 2
I : 14	FLIGHT.TIME : 135	AC.TYPE : 3	#AC : 2
I : 15	FLIGHT.TIME : 155	AC.TYPE : 1	#AC : 2
I : 16	FLIGHT.TIME : 156	AC.TYPE : 2	#AC : 2
I : 17	FLIGHT.TIME : 165	AC.TYPE : 3	#AC : 2
I : 18	FLIGHT.TIME : 185	AC.TYPE : 1	#AC : 4

I : 19	FLIGHT.TIME :	186	AC.TYPE :	2	#AC :	2
I : 20	FLIGHT.TIME :	195	AC.TYPE :	3	#AC :	2
I : 21	FLIGHT.TIME :	215	AC.TYPE :	1	#AC :	4
I : 22	FLIGHT.TIME :	216	AC.TYPE :	2	#AC :	2
I : 23	FLIGHT.TIME :	225	AC.TYPE :	3	#AC :	2
I : 24	FLIGHT.TIME :	245	AC.TYPE :	1	#AC :	2
I : 25	FLIGHT.TIME :	246	AC.TYPE :	2	#AC :	2
I : 26	FLIGHT.TIME :	255	AC.TYPE :	3	#AC :	2
I : 27	FLIGHT.TIME :	275	AC.TYPE :	1	#AC :	4
I : 28	FLIGHT.TIME :	276	AC.TYPE :	2	#AC :	2
I : 29	FLIGHT.TIME :	285	AC.TYPE :	3	#AC :	2
I : 30	FLIGHT.TIME :	305	AC.TYPE :	1	#AC :	4
I : 31	FLIGHT.TIME :	306	AC.TYPE :	2	#AC :	2
I : 32	FLIGHT.TIME :	315	AC.TYPE :	3	#AC :	2
I : 33	FLIGHT.TIME :	335	AC.TYPE :	1	#AC :	2
I : 34	FLIGHT.TIME :	336	AC.TYPE :	2	#AC :	2
I : 35	FLIGHT.TIME :	345	AC.TYPE :	3	#AC :	2

NUM.OPEN.SPOTS = 6

TIME: 0. EVENT: 8 FLT.TIME 20 FLT.TYPE 1
 THERE ARE 4 AC OF TYPE 1 IN THIS FLIGHT
 AC 1 AT 7 SCHED TO LAUNCH AT 20 AND RETURN AT 0
 AC : 1 COUNTER : 1
 NUM.OPEN.SPOTS = 5
 AC 2 AT 7 SCHED TO LAUNCH AT 20 AND RETURN AT 0
 AC : 2 COUNTER : 2
 NUM.OPEN.SPOTS = 4
 AC 3 AT 7 SCHED TO LAUNCH AT 20 AND RETURN AT 0
 AC : 3 COUNTER : 3
 NUM.OPEN.SPOTS = 3
 AC 4 AT 7 SCHED TO LAUNCH AT 20 AND RETURN AT 0
 AC : 4 COUNTER : 4
 NUM.OPEN.SPOTS = 2
 T.NUM.AC = 0 FLT.AC.NUM = 4 COUNTER = 4

TIME: 2.00 EVENT: 8 FLT.TIME 21 FLT.TYPE 2
 THERE ARE 2 AC OF TYPE 2 IN THIS FLIGHT
 AC 11 AT 8 SCHED TO LAUNCH AT 21 AND RETURN AT 0
 AC : 11 COUNTER : 1
 NUM.OPEN.SPOTS = 1
 AC 11 FILED IN TUG.Q
 AC 12 AT 8 SCHED TO LAUNCH AT 21 AND RETURN AT 0
 AC : 12 COUNTER : 2
 NUM.OPEN.SPOTS = 0
 AC 12 FILED IN TUG.Q
 T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

TIME: 3.77 EVENT: 3 AC.ID: 4 AC.DEST: 4
 AC 4 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.00

AC 11 REMOVED FROM TUG.Q

TIME: 4.00 EVENT: 8 FLT.TIME 35 FLT.TYPE 1
THERE ARE 4 AC OF TYPE 1 IN THIS FLIGHT
AC 5 AT 9 SCHED TO LAUNCH AT 35 AND RETURN AT 0
AC : 5 COUNTER : 1
AC 6 AT 9 SCHED TO LAUNCH AT 35 AND RETURN AT 0
AC : 6 COUNTER : 2
AC 7 AT 9 SCHED TO LAUNCH AT 35 AND RETURN AT 0
AC : 7 COUNTER : 3
AC 8 AT 9 SCHED TO LAUNCH AT 35 AND RETURN AT 0
AC : 8 COUNTER : 4
T.NUM.AC = 0 FLT.AC.NUM = 4 COUNTER = 4

TIME: 4.11 EVENT: 3 AC.ID: 1 AC.DEST: 1
AC 1 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.80
AC 12 REMOVED FROM TUG.Q

TIME: 4.37 EVENT: 3 AC.ID: 3 AC.DEST: 3
AC 3 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.80

TIME: 5.00 EVENT: 12
DELTA.UPDATE.TIME= 0. TIME-DELTA= 5.0000

TIME: 5.00 EVENT: 9
#OPEN SPOTS= 0 LAST DELTA UPDATE= 0 #DELTA AC= 6 N.SPOT.Q = 6
AC 5 WITH DEST 9 HAS PRIORITY 0.81 FUEL.STAT 0.96 FLYING.TIME 115.0
AC 6 WITH DEST 9 HAS PRIORITY 0.81 FUEL.STAT 0.96 FLYING.TIME 115.0
AC 7 WITH DEST 9 HAS PRIORITY 0.81 FUEL.STAT 0.96 FLYING.TIME 115.0
AC 8 WITH DEST 9 HAS PRIORITY 0.81 FUEL.STAT 0.96 FLYING.TIME 115.0
AC 13 WITH DEST 9 HAS PRIORITY 0.81 FUEL.STAT 0.97 FLYING.TIME 145.0
AC 14 WITH DEST 9 HAS PRIORITY 0.81 FUEL.STAT 0.97 FLYING.TIME 145.0
AC 14 AT 9 SCHED TO LAUNCH AT 36
AC 13 AT 9 SCHED TO LAUNCH AT 35
AC 8 AT 9 SCHED TO LAUNCH AT 35
AC 7 AT 9 SCHED TO LAUNCH AT 35
AC 6 AT 9 SCHED TO LAUNCH AT 35
AC 5 AT 9 SCHED TO LAUNCH AT 35

TIME: 5.44 EVENT: 3 AC.ID: 2 AC.DEST: 2
AC 2 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.80

TIME: 6.00 EVENT: 8 FLT.TIME 36 FLT.TYPE 2
THERE ARE 2 AC OF TYPE 2 IN THIS FLIGHT
AC 13 AT 9 SCHED TO LAUNCH AT 36 AND RETURN AT 0
AC : 13 COUNTER : 1
AC 14 AT 9 SCHED TO LAUNCH AT 36 AND RETURN AT 0
AC : 14 COUNTER : 2
T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

TIME: 7.33 EVENT: 3 AC.ID: 11 AC.DEST: 5
AC 11 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.30

TIME: 7.36 EVENT: 3 AC.ID: 12 AC.DEST: 6
AC 12 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.80

TIME: 8.00 EVENT: 8 FLT.TIME 45 FLT.TYPE 3
THERE ARE 2 AC OF TYPE 3 IN THIS FLIGHT
AC 17 AT 8 SCHED TO LAUNCH AT 45 AND RETURN AT 0
AC : 17 COUNTER : 1
AC 18 AT 8 SCHED TO LAUNCH AT 45 AND RETURN AT 0
AC : 18 COUNTER : 2
T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

TIME: 8.00 EVENT: 4 AC.ID: 17 AC.LDC: 8
AC 17 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.000
THIS AC LOADED PREVIOUSLY
AC.OP.STAT(AC) : 1.000
AT 8.00 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 17 AC.ID(ACE) : 17
#RDY AC IN FLIGHT = 1

TIME: 8.00 EVENT: 4 AC.ID: 18 AC.LDC: 8
AC 18 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.000
THIS AC LOADED PREVIOUSLY
AC.OP.STAT(AC) : 1.000
AT 8.00 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 18 AC.ID(ACE) : 18
#RDY AC IN FLIGHT = 2
AC 17 FILED IN SPOT.Q
AC 18 FILED IN SPOT.Q

TIME: 8.00 EVENT: 9
#OPEN SPOTS = 0 LAST DELTA UPDATE = 5 #DELTA AC = 6 N.SPOT.Q = 8

AC 5	WITH DEST 9	HAS PRIORITY 0.81	FUEL.STAT 0.93	FLYING.TIME 112.0
AC 6	WITH DEST 9	HAS PRIORITY 0.81	FUEL.STAT 0.93	FLYING.TIME 112.0
AC 7	WITH DEST 9	HAS PRIORITY 0.81	FUEL.STAT 0.93	FLYING.TIME 112.0
AC 8	WITH DEST 9	HAS PRIORITY 0.81	FUEL.STAT 0.93	FLYING.TIME 112.0
AC 13	WITH DEST 9	HAS PRIORITY 0.81	FUEL.STAT 0.95	FLYING.TIME 142.0
AC 14	WITH DEST 9	HAS PRIORITY 0.81	FUEL.STAT 0.95	FLYING.TIME 142.0
AC 14	AT 9	SCHED TO LAUNCH AT 36		
AC 13	AT 9	SCHED TO LAUNCH AT 36		
AC 8	AT 9	SCHED TO LAUNCH AT 35		
AC 7	AT 9	SCHED TO LAUNCH AT 35		
AC 6	AT 9	SCHED TO LAUNCH AT 35		
AC 5	AT 9	SCHED TO LAUNCH AT 35		
AC 17	AT 8	SCHED TO LAUNCH AT 45		
AC 18	AT 8	SCHED TO LAUNCH AT 45		

TIME: 8.04 EVENT: 16 AC.ID: 21 AC.LOC: 12
AC GOING TO DECK HAS ARRIVED AT THE ELEVATOR

TIME: 9.80 EVENT: 17 AC.ID: 21 AC.LOC: 12
HANGER.EQUIV = 1.0

TIME: 10.00 EVENT: 12
DELTA.UPDATE.TIME= 8.0000 TIME-DELTA= 2.0000

TIME: 11.66 EVENT: 6 AC.ID: 21 AC.LOC: 11
AC 21 WITH FUEL 0. AND LOAD 0. HAS OP.STAT 0.60
N.BONE.FWD= 2 N.BONE.AFT= 7 BONE.TOTAL= 9 NUM.OPEN.SPOTS= 0
N.LOAD.SET= 1

TIME: 11.71 EVENT: 4 AC.ID: 4 AC.LOC: 4
AC 4 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 11.71 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 4 AC.ID(ACE) : 4
*RDY AC IN FLIGHT = 1

TIME: 11.89 EVENT: 4 AC.ID: 1 AC.LOC: 1
AC 1 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 11.89 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 1 AC.ID(ACE) : 1
*RDY AC IN FLIGHT = 2

TIME: 12.30 EVENT: 4 AC.ID: 3 AC.LOC: 3
 AC 3 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.890
 AC.OP.STAT(AC) : 1.000
 AT 12.30 THERE ARE 5 FLIGHTS IN THE PLAN
 AC.ID(AC) : 3 AC.ID(ACE) : 3
 #RDY AC IN FLIGHT = 3

TIME: 12.88 EVENT: 4 AC.ID: 2 AC.LOC: 2
 AC 2 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
 AC.OP.STAT(AC) : 1.000
 AT 12.89 THERE ARE 5 FLIGHTS IN THE PLAN
 AC.ID(AC) : 2 AC.ID(ACE) : 2
 #RDY AC IN FLIGHT = 4
 FLIGHT.LAUNCH HAS BEEN SCHEDULED

TIME: 13.00 EVENT: 12
 DELTA.UPDATE.TIME= 8.0000 TIME-DELTA= 5.0000

TIME: 13.00 EVENT: 9
 #OPEN SPOTS= 0 LAST DELTA UPDATE= 8 #DELTA AC= 5 N.SPOT.Q = 8

AC 5 WITH DEST 9 HAS PRIORITY 0.82	FUEL.STAT 0.89	FLYING.TIME 107.0
AC 6 WITH DEST 9 HAS PRIORITY 0.82	FUEL.STAT 0.89	FLYING.TIME 107.0
AC 7 WITH DEST 9 HAS PRIORITY 0.82	FUEL.STAT 0.89	FLYING.TIME 107.0
AC 8 WITH DEST 9 HAS PRIORITY 0.82	FUEL.STAT 0.89	FLYING.TIME 107.0
AC 13 WITH DEST 9 HAS PRIORITY 0.82	FUEL.STAT 0.91	FLYING.TIME 137.0
AC 14 WITH DEST 9 HAS PRIORITY 0.82	FUEL.STAT 0.91	FLYING.TIME 137.0
AC 14 AT 9 SCHED TO LAUNCH AT 36		
AC 13 AT 9 SCHED TO LAUNCH AT 36		
AC 8 AT 9 SCHED TO LAUNCH AT 35		
AC 7 AT 9 SCHED TO LAUNCH AT 35		
AC 6 AT 9 SCHED TO LAUNCH AT 35		
AC 5 AT 9 SCHED TO LAUNCH AT 35		
AC 17 AT 8 SCHED TO LAUNCH AT 45		
AC 18 AT 9 SCHED TO LAUNCH AT 45		

TIME: 13.38 EVENT: 10 FLT.TIME 20 FLT.TYPE 1
 INTERVAL= 0.9 LAST.LAUNCH.TIME= 0. LAST.REC.TIME= 0.
 TIME: 13 AC 1 WILL LAUNCH IN 0.94 MINUTES
 TIME: 13 AC 2 WILL LAUNCH IN 1.44 MINUTES
 TIME: 13 AC 3 WILL LAUNCH IN 2.13 MINUTES
 TIME: 13 AC 4 WILL LAUNCH IN 2.81 MINUTES

FLIGHT 21 IS IN PLAN AND HAS	AC 11 AT LOCATION 5	WITH DEST 10
FLIGHT 21 IS IN PLAN AND HAS	AC 12 AT LOCATION 6	WITH DEST 10
FLIGHT 35 IS IN PLAN AND HAS	AC 5 AT LOCATION 9	WITH DEST 9
FLIGHT 35 IS IN PLAN AND HAS	AC 6 AT LOCATION 9	WITH DEST 9

FLIGHT 35 IS IN PLAN AND HAS AC 7 AT LOCATION 9 WITH DEST 9
 FLIGHT 35 IS IN PLAN AND HAS AC 8 AT LOCATION 9 WITH DEST 9
 FLIGHT 36 IS IN PLAN AND HAS AC 13 AT LOCATION 9 WITH DEST 9
 FLIGHT 36 IS IN PLAN AND HAS AC 14 AT LOCATION 9 WITH DEST 9
 FLIGHT 45 IS IN PLAN AND HAS AC 17 AT LOCATION 8 WITH DEST 0
 FLIGHT 45 IS IN PLAN AND HAS AC 18 AT LOCATION 9 WITH DEST 0

TIME: 14.31 EVENT: 1 AC.ID: 1 AC.LOC: 1
 AC 1 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
 #OPEN SPOTS = 1
 AC 1 LAUNCHES FROM 1 WITH DELAY -5.69
 AC 1 WILL ARRIVE TO DELTA AT 69 WITH PRIORITY 0.90 AND FLYING.TIME 57.2
 FLIGHT AT 20 WITH 5 AC LAUNCHES AC 1 AT 14.3

TIME: 14.82 EVENT: 1 AC.ID: 2 AC.LOC: 2
 AC 2 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
 #OPEN SPOTS = 2
 AC 2 LAUNCHES FROM 2 WITH DELAY -5.18
 AC 2 WILL ARRIVE TO DELTA AT 67 WITH PRIORITY 0.90 AND FLYING.TIME 59.5
 FLIGHT AT 20 WITH 5 AC LAUNCHES AC 2 AT 14.8

TIME: 15.51 EVENT: 1 AC.ID: 3 AC.LOC: 3
 AC 3 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
 #OPEN SPOTS = 3
 AC 3 LAUNCHES FROM 3 WITH DELAY -4.49
 AC 3 WILL ARRIVE TO DELTA AT 73 WITH PRIORITY 0.91 AND FLYING.TIME 54.3
 FLIGHT AT 20 WITH 5 AC LAUNCHES AC 3 AT 15.5

TIME: 16.19 EVENT: 1 AC.ID: 4 AC.LOC: 4
 AC 4 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
 #OPEN SPOTS = 4
 AC 4 LAUNCHES FROM 4 WITH DELAY -3.81
 AC 4 WILL ARRIVE TO DELTA AT 74 WITH PRIORITY 0.91 AND FLYING.TIME 53.4
 FLIGHT AT 20 WITH 5 AC LAUNCHES AC 4 AT 16.2

TIME: 16.50 EVENT: 9
 #OPEN SPOTS= 4. LAST DELTA UPDATE= 13 #DELTA AC= 5 N.SPOT.Q = 9
 AC 5 WITH DEST 9 HAS PRIORITY 0.93 FUEL.STAT 0.86 FLYING.TIME 103.5
 AC 6 WITH DEST 9 HAS PRIORITY 0.83 FUEL.STAT 0.86 FLYING.TIME 103.5
 AC 7 WITH DEST 9 HAS PRIORITY 0.93 FUEL.STAT 0.86 FLYING.TIME 103.5
 AC 8 WITH DEST 9 HAS PRIORITY 0.93 FUEL.STAT 0.86 FLYING.TIME 103.5
 AC 13 WITH DEST 9 HAS PRIORITY 0.82 FUEL.STAT 0.89 FLYING.TIME 133.5
 AC 14 WITH DEST 9 HAS PRIORITY 0.82 FUEL.STAT 0.89 FLYING.TIME 133.5
 AC 14 AT 9 SCHED TO LAUNCH AT 36
 AC 13 AT 9 SCHED TO LAUNCH AT 36

AC 8 AT 9 SCHED TO LAUNCH AT 35
 AC 7 AT 9 SCHED TO LAUNCH AT 35
 AC 6 AT 9 SCHED TO LAUNCH AT 35
 AC 5 AT 9 SCHED TO LAUNCH AT 35
 AC 17 AT 8 SCHED TO LAUNCH AT 45
 AC 13 AT 8 SCHED TO LAUNCH AT 45
 #OPEN SPOTS= 4 #SPOT.Q = 8 I = 6 SPOT(I) = 12
 #OPEN SPOTS= 4 #SPOT.Q = 8 I = 5 SPOT(I) = 11
 #OPEN SPOTS= 4 #SPOT.Q = 8 I = 4 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 21 OPEN SPOT IS 4
 #OPEN SPOTS= 4 #SPOT.Q = 8 I = 3 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 21 OPEN SPOT IS 3
 #OPEN SPOTS= 4 #SPOT.Q = 8 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 21 OPEN SPOT IS 2
 #OPEN SPOTS= 4 #SPOT.Q = 8 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 21 OPEN SPOT IS 1

TIME: 16.73 EVENT: 16 AC.ID: 22 AC.LOC: 12
 AC GOING TO DECK HAS ARRIVED AT THE ELEVATOR

TIME: 17.09 EVENT: 4 AC.ID: 12 AC.LOC: 6
 AC 12 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
 AC.OP.STAT(AC) : 1.000
 AT 17.09 THERE ARE 4 FLIGHTS IN THE PLAN
 AC.ID(AC) : 12 AC.ID(ACE) : 12
 #RDY AC IN FLIGHT = 1

TIME: 17.32 EVENT: 4 AC.ID: 11 AC.LOC: 5
 AC 11 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
 AC.OP.STAT(AC) : 1.000
 AT 17.32 THERE ARE 4 FLIGHTS IN THE PLAN
 AC.ID(AC) : 11 AC.ID(ACE) : 11
 #RDY AC IN FLIGHT = 2
 FLIGHT.LAUNCH HAS BEEN SCHEDULED

TIME: 17.69 EVENT: 5 AC.ID: 21 AC.LOC: 8
 AC 21 WITH FUEL 0. AND LOAD 0. HAS OP.STAT 0.60
 AC.OP.STAT(AC) : 0.80 FLYING.TIME = 75.0
 AV8 IS ALREADY BEING LOADED

TIME: 17.82 EVENT: 10 FLT.TIME 21 FLT.TYPE 2
 INTERVAL= 0.6 LAST.LAUNCH.TIME= 16.2 LAST.REC.TIME= 0.
 TIME: 18 AC 11 WILL LAUNCH IN 0.59 MINUTES
 TIME: 18 AC 12 WILL LAUNCH IN 1.05 MINUTES
 FLIGHT 35 IS IN PLAN AND HAS AC 5 AT LOCATION 9 WITH DEST 9

FLIGHT 35 IS IN PLAN AND HAS AC 6 AT LOCATION 9 WITH DEST 9
 FLIGHT 35 IS IN PLAN AND HAS AC 7 AT LOCATION 9 WITH DEST 9
 FLIGHT 35 IS IN PLAN AND HAS AC 8 AT LOCATION 9 WITH DEST 9
 FLIGHT 36 IS IN PLAN AND HAS AC 13 AT LOCATION 9 WITH DEST 9
 FLIGHT 36 IS IN PLAN AND HAS AC 14 AT LOCATION 9 WITH DEST 9
 FLIGHT 45 IS IN PLAN AND HAS AC 17 AT LOCATION 8 WITH DEST 0
 FLIGHT 45 IS IN PLAN AND HAS AC 18 AT LOCATION 8 WITH DEST 0

TIME: 17.85 EVENT: 17 AC.ID: 22 AC.LOC: 12
 HANGER.EQUIV = 0.

TIME: 18.00 EVENT: 12
 DELTA.UPDATE.TIME= 16.4995 TIME-DELTA= 1.5005

TIME: 18.41 EVENT: 1 AC.ID: 11 AC.LOC: 5
 AC 11 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
 #OPEN SPOTS = 5
 AC 11 LAUNCHES FROM 5 WITH DELAY -2.59
 AC 11 WILL ARRIVE TO DELTA AT 70 WITH PRIORITY 0.88 AND FLYING.TIME 90.5
 FLIGHT AT 21 WITH 3 AC LAUNCHES AC 11 AT 18.4

TIME: 18.87 EVENT: 1 AC.ID: 12 AC.LOC: 6
 AC 12 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
 #OPEN SPOTS = 6
 AC 12 LAUNCHES FROM 6 WITH DELAY -2.13
 AC 12 WILL ARRIVE TO DELTA AT 79 WITH PRIORITY 0.89 AND FLYING.TIME 81.3
 FLIGHT AT 21 WITH 3 AC LAUNCHES AC 12 AT 18.9

TIME: 19.28 EVENT: 9
 #OPEN SPOTS= 6 LAST DELTA UPDATE= 16 #DELTA AC= 6 N.SPOT.Q = 8
 AC 5 WITH DEST 9 HAS PRIORITY 0.83 FUEL.STAT 0.84 FLYING.TIME 100.7
 AC 6 WITH DEST 9 HAS PRIORITY 0.83 FUEL.STAT 0.84 FLYING.TIME 100.7
 AC 7 WITH DEST 9 HAS PRIORITY 0.83 FUEL.STAT 0.84 FLYING.TIME 100.7
 AC 8 WITH DEST 9 HAS PRIORITY 0.83 FUEL.STAT 0.84 FLYING.TIME 100.7
 AC 13 WITH DEST 9 HAS PRIORITY 0.83 FUEL.STAT 0.87 FLYING.TIME 130.7
 AC 14 WITH DEST 9 HAS PRIORITY 0.83 FUEL.STAT 0.87 FLYING.TIME 137.7
 AC 14 AT 9 SCHED TO LAUNCH AT 36
 AC 13 AT 9 SCHED TO LAUNCH AT 36
 AC 3 AT 9 SCHED TO LAUNCH AT 35
 AC 7 AT 9 SCHED TO LAUNCH AT 35
 AC 6 AT 9 SCHED TO LAUNCH AT 35
 AC 5 AT 9 SCHED TO LAUNCH AT 35
 AC 17 AT 8 SCHED TO LAUNCH AT 45
 AC 13 AT 8 SCHED TO LAUNCH AT 45
 #OPEN SPOTS= 6 #SPOT.Q = 3 I = 6 SPOT(I) = 0

AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 35 OPEN SPOT IS 6
 AC 14 COMPATIBLE TO GO TO SPOT 6 HAS LAUNCH.TIME= 36 AND FLYING.TIME= 131
 NUM.OPEN.SPOTS = 5
 AC 14 WILL RECOVER TO SPOT 6 IN 4.9 MINUTES
 #IN SPOT.Q 7 #OPEN SPOTS 5 I 6 SPOT(I)-14 AC.ID 14 AC.LOC 9
 #OPEN SPOTS= 5 #SPOT.Q = 7 I = 5 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 35 OPEN SPOT IS 5
 AC 13 COMPATIBLE TO GO TO SPOT 5 HAS LAUNCH.TIME= 36 AND FLYING.TIME= 131
 NUM.OPEN.SPOTS = 4
 AC 13 WILL RECOVER TO SPOT 5 IN 5.5 MINUTES
 #IN SPOT.Q 6 #OPEN SPOTS 4 I 5 SPOT(I)-13 AC.ID 13 AC.LOC 9
 #OPEN SPOTS= 4 #SPOT.Q = 6 I = 4 SPOT(I) = 1
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 35 OPEN SPOT IS 4
 AC 8 COMPATIBLE TO GO TO SPOT 4 HAS LAUNCH.TIME= 35 AND FLYING.TIME= 101
 NUM.OPEN.SPOTS = 3
 AC 8 WILL RECOVER TO SPOT 4 IN 6.1 MINUTES
 #IN SPOT.Q 5 #OPEN SPOTS 3 I 4 SPOT(I)-8 AC.ID 8 AC.LOC 9
 #OPEN SPOTS= 3 #SPOT.Q = 5 I = 3 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 35 OPEN SPOT IS 3
 AC 7 COMPATIBLE TO GO TO SPOT 3 HAS LAUNCH.TIME= 35 AND FLYING.TIME= 101
 NUM.OPEN.SPOTS = 2
 AC 7 WILL RECOVER TO SPOT 3 IN 6.7 MINUTES
 #IN SPOT.Q 4 #OPEN SPOTS 2 I 3 SPOT(I)-7 AC.ID 7 AC.LOC 9
 #OPEN SPOTS= 2 #SPOT.Q = 4 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 35 OPEN SPOT IS 2
 AC 6 COMPATIBLE TO GO TO SPOT 2 HAS LAUNCH.TIME= 35 AND FLYING.TIME= 101
 NUM.OPEN.SPOTS = 1
 AC 6 WILL RECOVER TO SPOT 2 IN 7.4 MINUTES
 #IN SPOT.Q 3 #OPEN SPOTS 1 I 2 SPOT(I)-6 AC.ID 6 AC.LOC 9
 #OPEN SPOTS= 1 #SPOT.Q = 3 I = 1 SPOT(I) = 1
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 35 OPEN SPOT IS 1
 AC 5 COMPATIBLE TO GO TO SPOT 1 HAS LAUNCH.TIME= 35 AND FLYING.TIME= 101
 NUM.OPEN.SPOTS = 0
 AC 5 WILL RECOVER TO SPOT 1 IN 7.9 MINUTES
 #IN SPOT.Q 2 #OPEN SPOTS 0 I 1 SPOT(I)-5 AC.ID 5 AC.LOC 9

TIME: 21.50 EVENT: 12
 DELTA.UPDATE.TIME= 19.2766 TIME-DELTA= 2.2229

TIME: 22.12 EVENT: 6 AC.ID: 22 AC.LOC: 11
 AC 22 WITH FUEL 0. AND LOAD 0. HAS OP.STAT 0.60
 N.BONE.FWD= 2 N.BONE.AFT= 8 BONE.TOTAL= 10 NUM.OPEN.SPOTS= 0
 N.LOAD.SET= 2

TIME: 22.58 EVENT: 18 FLT.TIME 27 FLT.TYPE 1
 THIS FLIGHT HAS LAUNCHED

TIME: 22.69 EVENT: 18 FLT.TIME 21 FLT.TYPE 2
THIS FLIGHT HAS LAUNCHED

TIME: 23.40 EVENT: 2 AC.ID: 14 AC.DEST: 6
AC 14 WITH FUEL 0.87 AND LOAD 0. HAS OP.STAT 1.00
TO TIME= 36 LOC= 6 N.SPOT.Q= 2 NUM.OPEN.SPOTS= 0 AV8 TO TIME= 45
REFUELER= 1 N.REFUELER.Q = 0

TIME: 24.19 EVENT: 2 AC.ID: 13 AC.DEST: 5
AC 13 WITH FUEL 0.87 AND LOAD 0. HAS OP.STAT 1.00
TO TIME= 36 LOC= 5 N.SPOT.Q= 2 NUM.OPEN.SPOTS= 0 AV8 TO TIME= 45
REFUELER= 2 N.REFUELER.Q = 0

TIME: 24.28 EVENT: 12
DELTA.UPDATE.TIME= 19.2766 TIME-DELTA= 5.0000

TIME: 24.28 EVENT: 9
#OPEN SPOTS= 2 LAST DELTA UPDATE= 19 #DELTA AC= 4 N.SPOT.Q = 2
AC 5 WITH DEST 1 HAS PRIORITY 0.84 FUEL.STAT 0.80 FLYING.TIME 95.7
AC 6 WITH DEST 2 HAS PRIORITY 0.84 FUEL.STAT 0.80 FLYING.TIME 95.7
AC 7 WITH DEST 3 HAS PRIORITY 0.84 FUEL.STAT 0.80 FLYING.TIME 95.7
AC 8 WITH DEST 4 HAS PRIORITY 0.84 FUEL.STAT 0.80 FLYING.TIME 95.7
AC 17 AT 3 SCHED TO LAUNCH AT 45
AC 18 AT 8 SCHED TO LAUNCH AT 45

TIME: 24.81 EVENT: 2 AC.ID: 8 AC.DEST: 4
AC 8 WITH FUEL 0.80 AND LOAD 0. HAS OP.STAT 1.00
TO TIME= 35 LOC= 4 N.SPOT.Q= 2 NUM.OPEN.SPOTS= 0 AV8 TO TIME= 45
REFUELER= 3 N.REFUELER.Q = 0

TIME: 24.94 EVENT: 5 AC.ID: 14 AC.LOC: 6
AC 14 WITH FUEL 0.87 AND LOAD 0. HAS OP.STAT 0.77
AC.OP.STAT(AC) : 0.80 FLYING.TIME = 150.0

TIME: 25.00 EVENT: 8 FLT.TIME 65 FLT.TYPE 1
THERE ARE 2 AC OF TYPE 1 IN THIS FLIGHT
AC 9 AT 7 SCHED TO LAUNCH AT 65 AND RETURN AT 0
AC : 9 COUNTER : 1
AC 9 FILED IN SPOT.Q
AC 10 AT 7 SCHED TO LAUNCH AT 65 AND RETURN AT 0
AC : 10 COUNTER : 2

AC 10 FILED IN SPOT.Q
T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

TIME: 25.36 EVENT: 2 AC.ID: 7 AC.DEST: 3
AC 7 WITH FUEL 0.80 AND LOAD 0. HAS OP.STAT 1.00
TO TIME= 35 LOC= 3 N.SPOT.Q= 4 NUM.OPEN.SPOTS= 0 AV8 TO TIME= 45
REFUELER= 3 N.REFUELER.Q = 0

TIME: 25.39 EVENT: 4 AC.ID: 21 AC.LOC: 8
AC 21 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 25.39 THERE ARE 4 FLIGHTS IN THE PLAN

TIME: 25.77 EVENT: 5 AC.ID: 13 AC.LOC: 5
AC 13 WITH FUEL 0.87 AND LOAD 0. HAS OP.STAT 0.77
AC.OP.STAT(AC) : 0.90 FLYING.TIME = 190.0

TIME: 25.93 EVENT: 2 AC.ID: 6 AC.DEST: 2
AC 6 WITH FUEL 0.80 AND LOAD 0. HAS OP.STAT 1.00
TO TIME= 35 LOC= 2 N.SPOT.Q= 4 NUM.OPEN.SPOTS= 0 AV8 TO TIME= 45
REFUELER= 3 N.REFUELER.Q = 0

TIME: 26.00 EVENT: 8 FLT.TIME 66 FLT.TYPE 2
THERE ARE 2 AC OF TYPE 2 IN THIS FLIGHT
AC 15 AT 8 SCHED TO LAUNCH AT 66 AND RETURN AT 0
AC : 15 COUNTER : 1
AC 15 FILED IN SPOT.Q
AC 16 AT 9 SCHED TO LAUNCH AT 66 AND RETURN AT 0
AC : 16 COUNTER : 2
AC 16 FILED IN SPOT.Q
T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

TIME: 26.70 EVENT: 2 AC.ID: 5 AC.DEST: 1
AC 5 WITH FUEL 0.80 AND LOAD 0. HAS OP.STAT 1.00
TO TIME= 35 LOC= 1 N.SPOT.Q= 6 NUM.OPEN.SPOTS= 0 AV8 TO TIME= 45
REFUELER= 4 N.REFUELER.Q = 0

TIME: 26.95 EVENT: 5 AC.ID: 9 AC.LOC: 4
AC 9 WITH FUEL 0.80 AND LOAD 0. HAS OP.STAT 0.76
AC.OP.STAT(AC) : 0.90 FLYING.TIME = 120.0

TIME: 27.27 EVENT: 5 AC.ID: 7 AC.LOC: 3
AC 7 WITH FUEL 0.80 AND LOAD 0. HAS OP.STAT 0.76
AC.OP.STAT(AC) : 0.30 FLYING.TIME = 120.0

TIME: 27.70 EVENT: 5 AC.ID: 6 AC.LOC: 2
AC 6 WITH FUEL 0.90 AND LOAD 0. HAS OP.STAT 0.76
AC.OP.STAT(AC) : 0.90 FLYING.TIME = 120.0

TIME: 27.84 EVENT: 5 AC.ID: 22 AC.LOC: 8
AC 22 WITH FUEL 0. AND LOAD 0. HAS OP.STAT 0.60
AC.OP.STAT(AC) : 0.80 FLYING.TIME = 75.0
AV9 IS ALREADY BEING LOADED

TIME: 29.28 EVENT: 12
DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 5.0000

TIME: 29.28 EVENT: 9
#OPEN SPOTS= 0 LAST DELTA UPDATE= 24 #DELTA AC= 3 N.SPOT.Q = 6
AC 17 AT 9 SCHED TO LAUNCH AT 45
AC 18 AT 9 SCHED TO LAUNCH AT 45
AC 9 AT 7 SCHED TO LAUNCH AT 65
AC 10 AT 7 SCHED TO LAUNCH AT 65
AC 15 AT 8 SCHED TO LAUNCH AT 66
AC 16 AT 8 SCHED TO LAUNCH AT 66

TIME: 29.31 EVENT: 5 AC.ID: 5 AC.LOC: 1
AC 5 WITH FUEL 0.80 AND LOAD 0. HAS OP.STAT 0.76
AC.OP.STAT(AC) : 0.80 FLYING.TIME = 120.0

TIME: 29.57 EVENT: 4 AC.ID: 7 AC.LOC: 3
AC 7 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 29.57 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 7 AC.ID(ACE) : 7
WRDY AC IN FLIGHT = 1

TIME: 30.68 EVENT: 4 AC.ID: 9 AC.LOC: 4
AC 9 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 1.000
AC.OP.STAT(AC) : 1.000

AT 30.68 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 8 AC.ID(ACE) : 8
#RDY AC IN FLIGHT = 2

TIME: 31.75 EVENT: 4 AC.ID: 6 AC.LOC: 2
AC 6 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 31.75 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 6 AC.ID(ACE) : 6
#RDY AC IN FLIGHT = 3

TIME: 33.09 EVENT: 4 AC.ID: 5 AC.LOC: 1
AC 5 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 33.09 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 5 AC.ID(ACE) : 5
#RDY AC IN FLIGHT = 4
FLIGHT.LAUNCH HAS BEEN SCHEDULED

TIME: 33.53 EVENT: 4 AC.ID: 13 AC.LOC: 5
AC 13 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 33.53 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 13 AC.ID(ACE) : 13
#RDY AC IN FLIGHT = 1

TIME: 33.59 EVENT: 10 FLT.TIME 35 FLT.TYPE 1
INTERVAL= 0.8 LAST.LAUNCH.TIME= 18.9 LAST.REC.TIME= 27.2
TIME: 34 AC 5 WILL LAUNCH IN 0.81 MINUTES
TIME: 34 AC 6 WILL LAUNCH IN 1.36 MINUTES
TIME: 34 AC 7 WILL LAUNCH IN 2.03 MINUTES
TIME: 34 AC 8 WILL LAUNCH IN 2.44 MINUTES
FLIGHT 36 IS IN PLAN AND HAS AC 13 AT LOCATION 5 WITH DEST 10
FLIGHT 36 IS IN PLAN AND HAS AC 14 AT LOCATION 6 WITH DEST 10
FLIGHT 43 IS IN PLAN AND HAS AC 17 AT LOCATION 9 WITH DEST 0
FLIGHT 43 IS IN PLAN AND HAS AC 18 AT LOCATION 9 WITH DEST 0
FLIGHT 65 IS IN PLAN AND HAS AC 9 AT LOCATION 7 WITH DEST 0
FLIGHT 65 IS IN PLAN AND HAS AC 10 AT LOCATION 7 WITH DEST 0
FLIGHT 66 IS IN PLAN AND HAS AC 15 AT LOCATION 3 WITH DEST 0
FLIGHT 66 IS IN PLAN AND HAS AC 16 AT LOCATION 3 WITH DEST 0

TIME: 33.68 EVENT: 4 AC.ID: 14 AC.LOC: 6
AC 14 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 33.68 THERE ARE 4 FLIGHTS IN THE PLAN

AC.ID(AC) : 14 AC.ID(ACE) : 14
#RDY AC IN FLIGHT = 2
FLIGHT-LAUNCH HAS BEEN SCHEDULED

TIME: 34.18 EVENT: 10 FLT.TIME 36 FLT.TYPE 2
INTERVAL= 0.9 LAST.LAUNCH.TIME= 36.0 LAST.REC.TIME= 27.2
TIME: 34 AC 13 WILL LAUNCH IN 2.85 MINUTES
TIME: 34 AC 14 WILL LAUNCH IN 3.53 MINUTES
FLIGHT 45 IS IN PLAN AND HAS AC 17 AT LOCATION 9 WITH DEST 0
FLIGHT 45 IS IN PLAN AND HAS AC 18 AT LOCATION 9 WITH DEST 0
FLIGHT 65 IS IN PLAN AND HAS AC 9 AT LOCATION 7 WITH DEST 0
FLIGHT 65 IS IN PLAN AND HAS AC 10 AT LOCATION 7 WITH DEST 0
FLIGHT 66 IS IN PLAN AND HAS AC 15 AT LOCATION 8 WITH DEST 0
FLIGHT 66 IS IN PLAN AND HAS AC 16 AT LOCATION 9 WITH DEST 0

TIME: 34.28 EVENT: 12
DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 10.0000

TIME: 34.28 EVENT: 9
#OPEN SPOTS= 3 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.C = 6
AC 17 AT 8 SCHED TO LAUNCH AT 45
AC 18 AT 8 SCHED TO LAUNCH AT 45
AC 9 AT 7 SCHED TO LAUNCH AT 65
AC 10 AT 7 SCHED TO LAUNCH AT 65
AC 15 AT 8 SCHED TO LAUNCH AT 66
AC 16 AT 9 SCHED TO LAUNCH AT 66

TIME: 34.40 EVENT: 1 AC.ID: 5 AC.LDC: 1
AC 5 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 1
AC 5 LAUNCHES FROM 1 WITH DELAY -0.60
AC 5 WILL ARRIVE TO DELTA AT 92 WITH PRIORITY 0.91 AND FLYING.TIME 53.2
FLIGHT AT 35 WITH 5 AC LAUNCHES AC 5 AT 34.4

TIME: 34.95 EVENT: 1 AC.ID: 6 AC.LDC: 2
AC 6 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 2
AC 6 LAUNCHES FROM 2 WITH DELAY -0.05
AC 6 WILL ARRIVE TO DELTA AT 92 WITH PRIORITY 0.92 AND FLYING.TIME 53.9
FLIGHT AT 35 WITH 5 AC LAUNCHES AC 6 AT 34.9

TIME: 35.00 EVENT: 3 FLT.TIME 75 FLT.TYPE 3
THERE ARE 2 AC OF TYPE 3 IN THIS FLIGHT

AC 19 AT 8 SCHED TO LAUNCH AT 75 AND RETURN AT 0
 AC : 19 COUNTER : 1
 AC 20 AT 8 SCHED TO LAUNCH AT 75 AND RETURN AT 0
 AC : 20 COUNTER : 2
 T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

TIME: 35.00 EVENT: 4 AC.ID: 19 AC.LOC: 8
 AC 19 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.000
 THIS AC LOADED PREVIOUSLY
 AC.OP.STAT(AC) : 1.000
 AT 35.00 THERE ARE 4 FLIGHTS IN THE PLAN
 AC.ID(AC) : 19 AC.ID(ACE) : 19
 #RDY AC IN FLIGHT = 1

TIME: 35.00 EVENT: 4 AC.ID: 20 AC.LOC: 8
 AC 20 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.000
 THIS AC LOADED PREVIOUSLY
 AC.OP.STAT(AC) : 1.000
 AT 35.00 THERE ARE 4 FLIGHTS IN THE PLAN
 AC.ID(AC) : 20 AC.ID(ACE) : 20
 #RDY AC IN FLIGHT = 2
 AC 19 FILED IN SPOT.Q
 AC 20 FILED IN SPOT.Q

TIME: 35.00 EVENT: 9
 #OPEN SPOTS= 2 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 8
 AC 17 AT 8 SCHED TO LAUNCH AT 45
 AC 18 AT 8 SCHED TO LAUNCH AT 45
 AC 9 AT 7 SCHED TO LAUNCH AT 65
 AC 10 AT 7 SCHED TO LAUNCH AT 65
 AC 15 AT 8 SCHED TO LAUNCH AT 66
 AC 16 AT 8 SCHED TO LAUNCH AT 66
 AC 19 AT 8 SCHED TO LAUNCH AT 75
 AC 20 AT 8 SCHED TO LAUNCH AT 75
 #OPEN SPOTS= 2 #SPOT.Q = 8 I = 6 SPOT(I) = 14
 #OPEN SPOTS= 2 #SPOT.Q = 8 I = 5 SPOT(I) = 13
 #OPEN SPOTS= 2 #SPOT.Q = 8 I = 4 SPOT(I) = 8
 #OPEN SPOTS= 2 #SPOT.Q = 8 I = 3 SPOT(I) = 7
 #OPEN SPOTS= 2 #SPOT.Q = 9 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 2
 AC 17 COMPATIBLE TO GO TO SPOT 2 HAS LAUNCH.TIME= 45 AND FLYING.TIME= 75
 AC = 17 WILL RESPOT TO 2
 NUM.OPEN.SPOTS = 1
 #IN SPOT.Q 7 #OPEN SPOTS 1 I 2 SPOT(I)=17 AC.ID 17 AC.LOC 9
 #OPEN SPOTS= 1 #SPOT.Q = 7 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 1
 AC 18 COMPATIBLE TO GO TO SPOT 1 HAS LAUNCH.TIME= 45 AND FLYING.TIME= 75
 AC = 18 WILL RESPOT TO 1
 NUM.OPEN.SPOTS = 0

#IN SPOT.Q 6 #OPEN SPOTS 0 I 1 SPOT(I)-18 AC.ID 18 AC.LOC 8

TIME: 35.62 EVENT: 1 AC.ID: 7 AC.LOC: 3
AC 7 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 1
AC 7 LAUNCHES FROM 3 WITH DELAY 0.62
AC 7 WILL ARRIVE TO DELTA AT 89 WITH PRIORITY 0.90 AND FLYING.TIME 58.3
FLIGHT AT 35 WITH 5 AC LAUNCHES AC 7 AT 35.6

TIME: 36.03 EVENT: 1 AC.ID: 8 AC.LOC: 4
AC 8 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 2
AC 8 LAUNCHES FROM 4 WITH DELAY 1.03
AC 8 WILL ARRIVE TO DELTA AT 94 WITH PRIORITY 0.91 AND FLYING.TIME 53.7
FLIGHT AT 35 WITH 5 AC LAUNCHES AC 8 AT 36.0

TIME: 36.40 EVENT: 9
#OPEN SPOTS= 2 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 6
AC 9 AT 7 SCHED TO LAUNCH AT 65
AC 10 AT 7 SCHED TO LAUNCH AT 65
AC 15 AT 8 SCHED TO LAUNCH AT 66
AC 16 AT 8 SCHED TO LAUNCH AT 66
AC 19 AT 8 SCHED TO LAUNCH AT 75
AC 20 AT 9 SCHED TO LAUNCH AT 75
#OPEN SPOTS= 2 #SPOT.Q = 6 I = 6 SPOT(I) = 14
#OPEN SPOTS= 2 #SPOT.Q = 6 I = 5 SPOT(I) = 13
#OPEN SPOTS= 2 #SPOT.Q = 6 I = 4 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 4
#OPEN SPOTS= 2 #SPOT.Q = 6 I = 3 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 3
#OPEN SPOTS= 2 #SPOT.Q = 6 I = 2 SPOT(I) = -17
#OPEN SPOTS= 2 #SPOT.Q = 6 I = 1 SPOT(I) = -18

TIME: 36.93 EVENT: 19 FLT.TIME 35 FLT.TYPE 1
THIS FLIGHT HAS LAUNCHED

TIME: 37.03 EVENT: 1 AC.ID: 13 AC.LOC: 5
AC 13 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 3
AC 13 LAUNCHES FROM 5 WITH DELAY 1.03
AC 13 WILL ARRIVE TO DELTA AT 92 WITH PRIORITY 0.38 AND FLYING.TIME 86.9
FLIGHT AT 36 WITH 3 AC LAUNCHES AC 13 AT 37.0

TIME: 37.09 EVENT: 3 AC.ID: 17 AC.DEST: 2
AC 17 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
AV9 17 ROY TO LAUNCH AT 37 FROM SPOT 2
AV9 FLIGHT SCHED TO LAUNCH AT 45 HAS 1 AV8'S ROY ON SPOTS

TIME: 37.24 EVENT: 18 FLT.TIME 36 FLT.TYPE 2
THIS FLIGHT HAS LAUNCHED

TIME: 37.71 EVENT: 1 AC.ID: 14 AC.LOC: 6
AC 14 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 4
AC 14 LAUNCHES FROM 6 WITH DELAY 1.71
AC 14 WILL ARRIVE TO DELTA AT 91 WITH PRIORITY 0.88 AND FLYING.TIME 88.5
FLIGHT AT 36 WITH 3 AC LAUNCHES AC 14 AT 37.7

TIME: 38.06 EVENT: 9
#OPEN SPOTS= 4 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 6
AC 9 AT 7 SCHED TO LAUNCH AT 65
AC 10 AT 7 SCHED TO LAUNCH AT 65
AC 13 AT 8 SCHED TO LAUNCH AT 66
AC 16 AT 8 SCHED TO LAUNCH AT 66
AC 19 AT 8 SCHED TO LAUNCH AT 75
AC 20 AT 8 SCHED TO LAUNCH AT 75
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 6 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 6
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 5 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 5
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 4 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 4
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 3 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 3
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 2 SPOT(I) = 17
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 1 SPOT(I) = -18

TIME: 38.25 EVENT: 4 AC.ID: 22 AC.LOC: 8
AC 22 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.900
AC.OP.STAT(AC) : 1.000
AT 38.25 THERE ARE 4 FLIGHTS IN THE PLAN

TIME: 39.20 EVENT: 3 AC.ID: 18 AC.DEST: 1
AC 18 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
AV8 18 ROY TO LAUNCH AT 39 FROM SPOT 1
AV3 FLIGHT SCHED TO LAUNCH AT 45 HAS 2 AV8'S ROY ON SPOTS
AV8 FLIGHT SCHE TO LAUNCH AT 45 HAS BEEN SENT TO FLIGHT.LAUNCH

TIME: 39.28 EVENT: 12
DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 15.0000

TIME: 39.28 EVENT: 9
#OPEN SPOTS= 4 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 6
AC 9 AT 7 SCHED TO LAUNCH AT 65
AC 10 AT 7 SCHED TO LAUNCH AT 65
AC 15 AT 8 SCHED TO LAUNCH AT 66
AC 16 AT 8 SCHED TO LAUNCH AT 66
AC 19 AT 8 SCHED TO LAUNCH AT 75
AC 20 AT 8 SCHED TO LAUNCH AT 75
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 6 SPOT(I) = 0
AV9.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 6
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 5 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 5
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 4 SPOT(I) = 0
AV8.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 4
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 3 SPOT(I) = 0
AV9.LAUNCH.TIME= 45 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 3
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 2 SPOT(I) = 17
#OPEN SPOTS= 4 #SPOT.Q = 6 I = 1 SPOT(I) = 18

TIME: 39.51 EVENT: 10 FLT.TIME 45 FLT.TYPE 3
INTERVAL= 0.9 LAST.LAUNCH.TIME= 37.7 LAST.REC.TIME= 27.2
TIME: 40 AC 17 WILL LAUNCH IN 0.88 MINUTES
TIME: 40 AC 18 WILL LAUNCH IN 1.42 MINUTES
FLIGHT 65 IS IN PLAN AND HAS AC 9 AT LOCATION 7 WITH DEST 0
FLIGHT 65 IS IN PLAN AND HAS AC 10 AT LOCATION 7 WITH DEST 0
FLIGHT 66 IS IN PLAN AND HAS AC 15 AT LOCATION 8 WITH DEST 0
FLIGHT 66 IS IN PLAN AND HAS AC 16 AT LOCATION 8 WITH DEST 0
FLIGHT 75 IS IN PLAN AND HAS AC 19 AT LOCATION 8 WITH DEST 0
FLIGHT 75 IS IN PLAN AND HAS AC 20 AT LOCATION 8 WITH DEST 0

TIME: 40.39 EVENT: 1 AC.ID: 17 AC.LCC: 2
AC 17 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 5
AC 17 LAUNCHES FROM 2 WITH DELAY -4.61
AC 17 WILL ARRIVE TO DELTA AT 73 WITH PRIORITY 0.90 AND FLYING.TIME 37.1
FLIGHT AT 45 WITH 3 AC LAUNCHES AC 17 AT 40.4

TIME: 40.93 EVENT: 1 AC.ID: 18 AC.LCC: 1
AC 13 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 5
AC 13 LAUNCHES FROM 1 WITH DELAY -4.07
AC 13 WILL ARRIVE TO DELTA AT 67 WITH PRIORITY 0.88 AND FLYING.TIME 45.4

FLIGHT AT 45 WITH 3 AC LAUNCHES AC 18 AT 40.9

TIME: 41.28 EVENT: 9
#OPEN SPOTS= 6 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 6
AC 9 AT 7 SCHED TO LAUNCH AT 65
AC 10 AT 7 SCHED TO LAUNCH AT 65
AC 15 AT 8 SCHED TO LAUNCH AT 66
AC 16 AT 9 SCHED TO LAUNCH AT 66
AC 19 AT 3 SCHED TO LAUNCH AT 75
AC 20 AT 8 SCHED TO LAUNCH AT 75
#OPEN SPOTS= 6 #SPOT.Q = 6 I = 6 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 6
AC 15 COMPATIBLE TO GO TO SPOT 6 HAS LAUNCH.TIME= 66 AND FLYING.TIME= 150
AC 15 WILL RESPOT TO SPOT 6
NUM.OPEN.SPOTS = 5
#IN SPOT.Q 5 #OPEN SPOTS 5 I 6 SPOT(I)-15 AC.ID 15 AC.LOC 8
#OPEN SPOTS= 5 #SPOT.Q = 5 I = 5 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 5
AC 16 COMPATIBLE TO GO TO SPOT 5 HAS LAUNCH.TIME= 66 AND FLYING.TIME= 150
AC 16 WILL RESPOT TO SPOT 5
NUM.OPEN.SPOTS = 4
#IN SPOT.Q 4 #OPEN SPOTS 4 I 5 SPOT(I)-16 AC.ID 16 AC.LOC 8
#OPEN SPOTS= 4 #SPOT.Q = 4 I = 4 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 4
AC 9 COMPATIBLE TO GO TO SPOT 4 HAS LAUNCH.TIME= 65 AND FLYING.TIME= 120
AC 9 WILL RESPOT TO SPOT 4
NUM.OPEN.SPOTS = 3
#IN SPOT.Q 3 #OPEN SPOTS 3 I 4 SPOT(I)-9 AC.ID 9 AC.LOC 7
#OPEN SPOTS= 3 #SPOT.Q = 3 I = 3 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 3
AC 10 COMPATIBLE TO GO TO SPOT 3 HAS LAUNCH.TIME= 65 AND FLYING.TIME= 120
AC 10 WILL RESPOT TO SPOT 3
NUM.OPEN.SPOTS = 2
#IN SPOT.Q 2 #OPEN SPOTS 2 I 3 SPOT(I)-10 AC.ID 10 AC.LOC 7
#OPEN SPOTS= 2 #SPOT.Q = 2 I = 2 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 2
#OPEN SPOTS= 2 #SPOT.Q = 2 I = 1 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 1

TIME: 43.22 EVENT: 3 AC.ID: 15 AC.DEST: 6
AC 15 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.90

TIME: 44.28 EVENT: 12
DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 20.0000

TIME: 44.28 EVENT: 9
#OPEN SPOTS= 2 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 2

AC 19 AT 8 SCHED TO LAUNCH AT 75
 AC 20 AT 8 SCHED TO LAUNCH AT 75
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 6 SPOT(I) = 15
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 5 SPOT(I) = -16
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 4 SPOT(I) = -9
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 3 SPOT(I) = -10
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 2
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 1

TIME: 44.38 EVENT: 3 AC.ID: 10 AC.DEST: 3
 AC 10 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 2.30

TIME: 44.66 EVENT: 3 AC.ID: 16 AC.DEST: 5
 AC 16 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.80

TIME: 45.48 EVENT: 3 AC.ID: 9 AC.DEST: 4
 AC 9 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.80

TIME: 46.80 EVENT: 18 FLT.TIME 45 FLT.TYPE 3
 THIS FLIGHT HAS LAUNCHED

TIME: 49.28 EVENT: 12
 DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 25.0000

TIME: 49.29 EVENT: 9
 #OPEN SPOTS= 2 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 2
 AC 19 AT 8 SCHED TO LAUNCH AT 75
 AC 20 AT 8 SCHED TO LAUNCH AT 75
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 6 SPOT(I) = 15
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 5 SPOT(I) = 16
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 4 SPOT(I) = 9
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 3 SPOT(I) = 10
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 2
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 1

TIME: 54.28 EVENT: 12
 DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 30.0000

TIME: 54.29 EVENT: 9
 #OPEN SPOTS= 2 LAST DELTA UPDATE= 24 #DELTA AC= 3 N.SPOT.Q = 2
 AC 19 AT 8 SCHED TO LAUNCH AT 75
 AC 20 AT 8 SCHED TO LAUNCH AT 75
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 6 SPOT(I) = 15
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 5 SPOT(I) = 16
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 4 SPOT(I) = 9
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 3 SPOT(I) = 10
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 2
 AC 19 COMPATIBLE TO GO TO SPOT 2 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 #IN SPOT.Q 2 #OPEN SPOTS 2 I 2 SPOT(I) 0 AC.ID 19 AC.LOC 8
 #OPEN SPOTS= 2 #SPOT.Q = 2 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 65 OPEN SPOT IS 1
 AC 19 COMPATIBLE TO GO TO SPOT 1 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 #IN SPOT.Q 2 #OPEN SPOTS 2 I 1 SPOT(I) 0 AC.ID 19 AC.LOC 8

TIME: 55.00 EVENT: 8 FLT.TIME 95 FLT.TYPE 1
 THERE ARE 4 AC OF TYPE 1 IN THIS FLIGHT
 AC 1 AT 10 SCHED TO LAUNCH AT 95 AND RETURN AT 69
 AC : 1 COUNTER : 1
 AC 2 AT 10 SCHED TO LAUNCH AT 95 AND RETURN AT 67
 AC : 2 COUNTER : 2
 AC 3 AT 10 SCHED TO LAUNCH AT 95 AND RETURN AT 73
 AC : 3 COUNTER : 3
 AC 4 AT 10 SCHED TO LAUNCH AT 95 AND RETURN AT 74
 AC : 4 COUNTER : 4
 T.NUM.AC = 0 FLT.AC.NUM = 4 COUNTER = 4

TIME: 56.00 EVENT: 8 FLT.TIME 96 FLT.TYPE 2
 THERE ARE 2 AC OF TYPE 2 IN THIS FLIGHT
 AC 11 AT 10 SCHED TO LAUNCH AT 96 AND RETURN AT 70
 AC : 11 COUNTER : 1
 AC 12 AT 10 SCHED TO LAUNCH AT 96 AND RETURN AT 79
 AC : 12 COUNTER : 2
 T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

TIME: 56.97 EVENT: 4 AC.ID: 9 AC.LOC: 4
 AC 9 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
 AC.OP.STAT(AC) : 1.000
 AT 56.97 THERE ARE 5 FLIGHTS IN THE PLAN
 AC.ID(AC) : 9 AC.ID(ACE) : 9
 #RDY AC IN FLIGHT = 1

TIME: 57.37 EVENT: 4 AC.ID: 10 AC.LOC: 3
AC 10 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 57.37 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 10 AC.ID(ACE) : 10
#RDY AC IN FLIGHT = 2
FLIGHT.LAUNCH HAS BEEN SCHEDULED

TIME: 57.87 EVENT: 10 FLT.TIME 65 FLT.TYPE 1
INTERVAL= 0.6 LAST.LAUNCH.TIME= 40.9 LAST.REG.TIME= 27.2
TIME: 58 AC 10 WILL LAUNCH IN 0.64 MINUTES
TIME: 58 AC 9 WILL LAUNCH IN 1.14 MINUTES
FLIGHT 66 IS IN PLAN AND HAS AC 15 AT LOCATION 6 WITH DEST 10
FLIGHT 66 IS IN PLAN AND HAS AC 16 AT LOCATION 5 WITH DEST 10
FLIGHT 75 IS IN PLAN AND HAS AC 19 AT LOCATION 9 WITH DEST 0
FLIGHT 75 IS IN PLAN AND HAS AC 20 AT LOCATION 9 WITH DEST 0
FLIGHT 95 IS IN PLAN AND HAS AC 1 AT LOCATION 10 WITH DEST 9
FLIGHT 95 IS IN PLAN AND HAS AC 2 AT LOCATION 10 WITH DEST 9
FLIGHT 95 IS IN PLAN AND HAS AC 3 AT LOCATION 10 WITH DEST 9
FLIGHT 95 IS IN PLAN AND HAS AC 4 AT LOCATION 10 WITH DEST 9
FLIGHT 96 IS IN PLAN AND HAS AC 11 AT LOCATION 10 WITH DEST 9
FLIGHT 96 IS IN PLAN AND HAS AC 12 AT LOCATION 10 WITH DEST 9

TIME: 58.51 EVENT: 1 AC.ID: 10 AC.LOC: 3
AC 10 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 3
AC 10 LAUNCHES FROM 3 WITH DELAY -6.49
AC 10 WILL ARRIVE TO DELTA AT 117 WITH PRIORITY 0.91 AND FLYING.TIME 52.4
FLIGHT AT 65 WITH 3 AC LAUNCHES AC 10 AT 58.5

TIME: 59.01 EVENT: 1 AC.ID: 9 AC.LOC: 4
AC 9 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
#OPEN SPOTS = 4
AC 9 LAUNCHES FROM 4 WITH DELAY -5.99
AC 9 WILL ARRIVE TO DELTA AT 119 WITH PRIORITY 0.92 AND FLYING.TIME 51.0
FLIGHT AT 65 WITH 3 AC LAUNCHES AC 9 AT 59.0

TIME: 59.28 EVENT: 12
DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 35.0000

TIME: 59.28 EVENT: 9
#OPEN SPOTS= 4 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 2
AC 19 AT 8 SCHED TO LAUNCH AT 75
AC 20 AT 8 SCHED TO LAUNCH AT 75
#OPEN SPOTS= 4 #SPOT.Q = 2 I = 5 SPOT(I) = 15

#OPEN SPOTS= 4 #SPOT.Q = 2 I = 5 SPOT(I) = 16
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 4 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 4
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 3 SPOT(I) = 0
 AV9.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 3
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 2 SPOT(I) = 0
 AV9.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 2
 AC 19 COMPATIBLE TO GO TO SPOT 2 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 #IN SPOT.Q 2 #OPEN SPOTS 4 I 2 SPOT(I) 0 AC.ID 19 AC.LOC 8
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 1
 AC 19 COMPATIBLE TO GO TO SPOT 1 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 #IN SPOT.Q 2 #OPEN SPOTS 4 I 1 SPOT(I) 0 AC.ID 19 AC.LOC 8

TIME: 59.57 EVENT: 9
 #OPEN SPOTS= 4 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 2
 AC 19 AT 8 SCHED TO LAUNCH AT 75
 AC 20 AT 8 SCHED TO LAUNCH AT 75
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 6 SPOT(I) = 15
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 5 SPOT(I) = 16
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 4 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 4
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 3 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 3
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 2
 AC 19 COMPATIBLE TO GO TO SPOT 2 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 #IN SPOT.Q 2 #OPEN SPOTS 4 I 2 SPOT(I) 0 AC.ID 19 AC.LOC 8
 #OPEN SPOTS= 4 #SPOT.Q = 2 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 1
 AC 19 COMPATIBLE TO GO TO SPOT 1 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 #IN SPOT.Q 2 #OPEN SPOTS 4 I 1 SPOT(I) 0 AC.ID 19 AC.LOC 8

TIME: 62.65 EVENT: 4 AC.ID: 15 AC.LOC: 6
 AC 15 WITH FUEL 1.00 AND LOAD 0. HAS DP.STAT 0.300
 AC.DP.STAT(AC) : 1.000
 AT 62.65 THERE ARE 4 FLIGHTS IN THE PLAN
 AC.ID(AC) : 15 AC.ID(ACE) : 15
 #RDY AC IN FLIGHT = 1

TIME: 64.28 EVENT: 12
 DELTA.UPDATE.TIME= 24.2766 TIME-DELTA= 40.0000

TIME: 64.28 EVENT: 9
 #OPEN SPOTS= 4 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 2
 AC 19 AT 8 SCHED TO LAUNCH AT 75
 AC 20 AT 8 SCHED TO LAUNCH AT 75

```

#OPEN SPOTS= 4 #SPOT.Q = 2 I = 6 SPOT(I) = 15
#OPEN SPOTS= 4 #SPOT.Q = 2 I = 5 SPOT(I) = 16
#OPEN SPOTS= 4 #SPOT.Q = 2 I = 4 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 4
#OPEN SPOTS= 4 #SPOT.Q = 2 I = 3 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 3
#OPEN SPOTS= 4 #SPOT.Q = 2 I = 2 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 2
AC 19 COMPATIBLE TO GO TO SPOT 2 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
#IN SPOT.Q 2 #OPEN SPOTS 4 I 2 SPOT(I) 0 AC.ID 19 AC.LOC 8
#OPEN SPOTS= 4 #SPOT.Q = 2 I = 1 SPOT(I) = 0
AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 1
AC 19 COMPATIBLE TO GO TO SPOT 1 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
#IN SPOT.Q 2 #OPEN SPOTS 4 I 1 SPOT(I) 0 AC.ID 19 AC.LOC 8

```

```

TIME: 64.65 EVENT: 4 AC.ID: 16 AC.LOC: 5
AC 16 WITH FUEL 1.00 AND LOAD 0. HAS OP.STAT 0.800
AC.OP.STAT(AC) : 1.000
AT 64.65 THERE ARE 4 FLIGHTS IN THE PLAN
AC.ID(AC) : 16 AC.ID(ACE) : 16
#RDY AC IN FLIGHT = 2
FLIGHT.LAUNCH HAS BEEN SCHEDULED

```

```

TIME: 65.00 EVENT: 8 FLT.TIME 105 FLT.TYPE 3
THERE ARE 2 AC OF TYPE 3 IN THIS FLIGHT
AC 21 AT 8 SCHED TO LAUNCH AT 105 AND RETURN AT 0
AC : 21 COUNTER : 1
AC 22 AT 8 SCHED TO LAUNCH AT 105 AND RETURN AT 0
AC : 22 COUNTER : 2
T.NUM.AC = 0 FLT.AC.NUM = 2 COUNTER = 2

```

```

TIME: 65.00 EVENT: 4 AC.ID: 21 AC.LOC: 9
AC 21 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.000
THIS AC LOADED PREVIOUSLY
AC.OP.STAT(AC) : 1.000
AT 65.00 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 21 AC.ID(ACE) : 21
#RDY AC IN FLIGHT = 1

```

```

TIME: 65.00 EVENT: 4 AC.ID: 22 AC.LOC: 9
AC 22 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.000
THIS AC LOADED PREVIOUSLY
AC.OP.STAT(AC) : 1.000
AT 65.00 THERE ARE 5 FLIGHTS IN THE PLAN
AC.ID(AC) : 22 AC.ID(ACE) : 22
#RDY AC IN FLIGHT = 2
AC 21 FILED IN SPOT.Q

```

AC 22 FILED IN SPOT.Q

TIME: 65.00 EVENT: 9
 #OPEN SPOTS= 4 LAST DELTA UPDATE= 24 #DELTA AC= 0 N.SPOT.Q = 4
 AC 19 AT 8 SCHED TO LAUNCH AT 75
 AC 20 AT 8 SCHED TO LAUNCH AT 75
 AC 21 AT 8 SCHED TO LAUNCH AT 105
 AC 22 AT 8 SCHED TO LAUNCH AT 105
 #OPEN SPOTS= 4 #SPOT.Q = 4 I = 6 SPOT(I) = 15
 #OPEN SPOTS= 4 #SPOT.Q = 4 I = 5 SPOT(I) = 16
 #OPEN SPOTS= 4 #SPOT.Q = 4 I = 4 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 4
 #OPEN SPOTS= 4 #SPOT.Q = 4 I = 3 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 3
 #OPEN SPOTS= 4 #SPOT.Q = 4 I = 2 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 2
 AC 19 COMPATIBLE TO GO TO SPOT 2 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 AC = 19 WILL RESPOT TO 2
 NUM.OPEN.SPOTS = 3
 #IN SPOT.Q 3 #OPEN SPOTS 3 I 2 SPOT(I)-19 AC.ID 19 AC.LOC 8
 #OPEN SPOTS= 3 #SPOT.Q = 3 I = 1 SPOT(I) = 0
 AV8.LAUNCH.TIME= 75 HELD.LAUNCH.TIME= 66 OPEN SPOT IS 1
 AC 20 COMPATIBLE TO GO TO SPOT 1 HAS LAUNCH.TIME= 75 AND FLYING.TIME= 75
 AC = 20 WILL RESPOT TO 1
 NUM.OPEN.SPOTS = 2
 #IN SPOT.Q 2 #OPEN SPOTS 2 I 1 SPOT(I)-20 AC.ID 20 AC.LOC 8

TIME: 65.15 EVENT: 10 FLT.TIME 66 FLT.TYPE 2
 INTERVAL= 0.8 LAST.LAUNCH.TIME= 59.0 LAST.REC.TIME= 27.2
 TIME: 65 AC 16 WILL LAUNCH IN 0.75 MINUTES
 TIME: 65 AC 15 WILL LAUNCH IN 1.35 MINUTES
 FLIGHT 75 IS IN PLAN AND HAS AC 19 AT LOCATION 8 WITH DEST 2
 FLIGHT 75 IS IN PLAN AND HAS AC 20 AT LOCATION 8 WITH DEST 1
 FLIGHT 95 IS IN PLAN AND HAS AC 1 AT LOCATION 10 WITH DEST 9
 FLIGHT 95 IS IN PLAN AND HAS AC 2 AT LOCATION 10 WITH DEST 9
 FLIGHT 95 IS IN PLAN AND HAS AC 3 AT LOCATION 10 WITH DEST 9
 FLIGHT 95 IS IN PLAN AND HAS AC 4 AT LOCATION 10 WITH DEST 9
 FLIGHT 96 IS IN PLAN AND HAS AC 11 AT LOCATION 10 WITH DEST 9
 FLIGHT 96 IS IN PLAN AND HAS AC 12 AT LOCATION 10 WITH DEST 9
 FLIGHT 105 IS IN PLAN AND HAS AC 21 AT LOCATION 8 WITH DEST 8
 FLIGHT 105 IS IN PLAN AND HAS AC 22 AT LOCATION 9 WITH DEST 8

TIME: 65.90 EVENT: 1 AC.ID: 16 AC.LOC: 5
 AC 16 WITH FUEL 1.00 AND LOAD 1.00 HAS OP.STAT 1.00
 #OPEN SPOTS = 3
 AC 16 LAUNCHES FROM 5 WITH DELAY -0.10
 AC 16 WILL ARRIVE TO DELTA AT 121 WITH PRIORITY 0.88 AND FLYING.TIME 86.3
 FLIGHT AT 66 WITH 3 AC LAUNCHES AC 16 AT 65.9

CNA PROFESSIONAL PAPER INDEX¹

PP 407²

Laird, Robbin F. *The French Strategic Dilemma*. 22 pp., Nov 1984

PP 415

Mizrachi, Maurice M. *Can Authoritative Studies Be Trusted?* 2 pp., Jun 1984

PP 416

Jondrow, James M., and Levy, Robert A. *The Displacement of Local Spending for Pollution Control by Federal Construction Grants*, 6 pp., Jun 1984 (Reprinted from *American Economic Review*, May 1984)

PP 418

Reslock, Patricia A. *The Care and Feeding of Magnetic Tapes*, 7 pp., Jul 1984

PP 420

Weiss, Kenneth G. *The War for the Falklands: A Chronology*, 32 pp., Aug 1982

PP 422

Quester, Aline, and Marcus, Alan. *An Evaluation of The Effectiveness of Classroom and On the Job Training*, 35 pp., Dec 1984. (Presented at the Symposium on Training Effectiveness, NATO Defense Research Group, Brussels, 7-9 January 1985)

PP 423

Dismukes, N. Bradford, and Weiss, Kenneth G. *MARE MOSSO: The Mediterranean Theater*, 26 pp., Nov 1984. (Presented at the Seapower Conference, Washington, D.C., 26-27 November 1984)

PP 424

Berg, Dr. Robert M., *The CNA Ordnance Programming Model and Methodology*, 27 pp., Oct 1984. (Presented at the ORSA-MAS/MDRS Symposium, Washington, Aug 1984)

PP 425

Horowitz, Stanley A., and Angier, Bruce N. *Costs and Benefits of Training and Experience*, 18 pp., Jan 1985. (Presented at the Symposium on Training Effectiveness, NATO Defense Research Group, Brussels, 7-9 January 1985)

PP 427

Cavalluzzo, Linda C. *Op Tempo and Training Effectiveness*, 19 pp., Dec 1984. (Presented at the Symposium on Training Effectiveness, NATO Defense Research Group, Brussels, 7-9 January 1985)

PP 428

Matthes, Greg, Cdr., USN and Evanovich, Peter *Force Levels, Readiness, and Capability*, 24 pp., Nov 1984. (Presented at the ORSA-TIMS 26-28 November Meeting, Washington, D.C.)

PP 429

Perla, Peter P. and Barrett, Raymond T., LCdr., USN, *Wargaming and Its Uses*, 13 pp., Nov 1984. (Published in the *Naval War College Review*, XXXVIII, No. 5 / Sequence 311, September-October 1985)

PP 430

Goldberg, Matthew S. *The Relationship Between Material Failures And Flight Hours: Statistical Considerations*, 18 pp., Jan 1985

PP 431

McConnell, James M. *A Possible Change in Soviet Views on the Prospects for Anti-Submarine Warfare*, 19 pp., Jan 1985

PP 432

Marcus, Alan J. and Curran, Lawrence E., Cdr., USN, *The Use of Flight Simulators in Measuring and Improving Training Effectiveness*, 29 pp., Jan 1985. (Presented at the Symposium on Training Effectiveness, NATO Defense Research Group, Brussels, 7-9 January 1985)

PP 433

Quester, Aline O. and Lockman, Robert F. *The All Volunteer Force: Outlook for the Eighties and Nineties*, 20 pp., Mar 1984. (To be published in *Armed Forces and Society*, 1985)

PP 435

Levine, Daniel B. and Jondrow, James M. *Readiness or Resources: Which Comes First?* 12 pp., Mar 1985

PP 436

Goldberg, Matthew S. *Logit Specification Tests Using Grouped Data*, 26 pp., Jan 1985

¹ CNA Professional Papers with an AD number may be obtained from the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22151. Other papers are available from the Management Information Office, Center for Naval Analyses, 4401 Ford Avenue, Alexandria, Virginia 22302-1268. An index of selected publications is also available, on request. The index includes a listing of professional papers, with abstracts, issued from 1969 to December 1983.

² Listings for Professional Papers issued prior to PP 407 can be found in *Index of Selected Publications (through December 1983)*, March 1984.

CNA PROFESSIONAL PAPER INDEX (Continued)

PP 438

Fletcher, Jean W. *Supply Problems in the Naval Reserve*, 14 pp., Feb 1986. (Presented at the Third Annual Mobilization Conference, Industrial College of the Armed Forces, National Defense University)

PP 440

Bell, Jr., Thomas D. *The Center for Naval Analyses: Past, Present, and Future*, 12 pp., Aug 1985

PP 441

Schneider, George R. *Implications of the Strategic Defense Initiative for the ABM Treaty*, 13 pp., Feb 1986. (Published in *Survival*, September-October 1985)

PP 442

Berg, Robert, Dennis, Richard, and Jondrow, James. *Price Analysis and the Effects of Competition*, 23 pp., Sep 1985. (Presented at the Association for Public Policy Analysis and Management - The Annual Research Conference, Shoreham Hotel, Washington, D.C., 25 October 1985)

PP 443

FitzGerald, Mary C., *Marshal Ogarkov on Modern War: 1977-1985*, 65 pp., Mar 1986

PP 445

Kober, Stanley. *Strategic Defense, Deterrence, and Arms Control*, 23 pp., Aug 1986. (Published in *The Washington Quarterly*, Winter 1986)

PP 446

Mayberry, Paul W. and Maier, Milton H., *Towards Justifying Enlistment Standards: Linking Input Characteristics to Job Performance*, 11 pp., Oct 1986. (Paper to be presented at 1986 American Psychological Association symposium entitled "Setting Standards in Performance Measurement")

PP 448

Cymrot, Donald J., *Military Retirement and Social Security - A Comparative Analysis*, 28 pp., Oct 1986

PP 449

Richardson, Henry R., *Search Theory*, 10 pp., Apr 1986

PP 451

FitzGerald, Mary C., *The Soviet Leadership on Nuclear War*, 40 pp., Apr 1987

PP 452

Mayberry, Paul W., *Issues in the Development of a Competency Scale: Implications for Linking Job Performance and Aptitude*, 22 pp., Apr 1987

PP 453

Dismukes, Bradford, *Strategic ASW And The Conventional Defense Of Europe*, 26 pp., Apr 1987

PP 454

Maier, Milton, *Marine Corps Project To Validate The ASVAB Against Job Performance*, 14 pp., May 1987

PP 456

Gates, Stephen, *Simulation and Analysis of Flight Deck Operations on an LHA*, 81 pp., Jun 1987

END

8-87

DTIC