Airport Simulation

Topic(s): Code Management/Reusability
Date Assigned: 10/11/10
Date Due: 10/27/10
Points: 50

For this assignment, you are to write a C program that will simulate an airport landing and takeoff pattern. The airport has 3 runways: runway1, runway2, and runway3. There is a holding pattern list into which arriving planes are entered. These planes have an associated integer value giving the number of time units the plane can remain in the list before it must land (because of low fuel). There is also a list for takeoffs into which departing planes are entered.

At each time step, 0-3 planes may desire to land (entering the holding pattern list), and 0-3 planes may desire to takeoff (entering the takeoff list). Each runway can handle one takeoff or landing during each time step. At each time step the following operations must be performed in the order listed below:

1. Read a line of data from the file **airport.txt** which is to be placed in a folder called Testfiles in your project folder. Each line has the following integer information: the number of planes to be added to the takeoff list; the number of planes to be added to the holding pattern list; the units of flying time for planes added to the holding pattern list.
2. Enter the new planes into their appropriate lists in the order read.
3. Decrement each holding plane's time (fuel) remaining. Increment takeoff plane's waiting time.
4. Those planes in the holding list with no time remaining (a value of 0) must be assigned runways for landing. When all runways become exhausted those planes yet to land with a fuel value of 0 are considered to have crashed.
5. The remaining runways are sequentially scheduled from the larger list, choosing from the holding pattern list if they are the same size. Note: the holding pattern list and takeoff list are first-in-first-out with the holding list being a priority list where planes with a value of 0 need a runway before anyone else.
6. Finally, increment the time and print the results for that time step.

Your output should clearly indicate what occurs at each time step: what was input, what happened on each runway, any crashes, and the resulting list lengths. At the end of the simulation (determined by the end-of-file on the data) , you are to summarize the results by providing the following:

1. the average takeoff waiting time
2. the average landing waiting time (includes those planes crashing)
3. the average flying time remaining on landing (includes those planes landing and crashing)
4. the number of planes landing with no fuel reserve (includes those planes crashing)
5. the number of crashes

Use the following format exactly for printing your output. Print a screen's worth of information at a time (including the heading each time) and allow the user to hit return before showing the next screen. The first line of numbers 12345…. is for your reference and not to be printed.

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890

     |           Planes Added             |       Runways       |    List  Lengths
Time | Takeoff  Landing (Fuel Remaining) |  1   2   3  Crash | Takeoff  Landing
---- | -------  ------------------------ | --- --- --- ----- | -------  -------
  1  |       2        3      (7   5   9) |  L   L   T    0   |    1        1
  2  |       2        3      (6   7   5) |  L   L   T    0   |    2        2
  3  |       2        3      (2   9   4) |  L   L   T    0   |    3        3
  4  |       3        2      (1   5   -) |  P   P   T    0   |    5        3
  5  |       2        2      (6   2   -) |  T   T   L    0   |    5        4
```

Print the summary information 1. through 5. on a screen by itself. Each summary result is to be a float and printed to two decimal places.

Lists must be implemented using your static list implementation in C from assignment #4.

You are to write a driver called airportsimulation.c that is the main driver module. You can implement as many modules as make sense for this project, but you also need to include the StaticList module that you completed from the last assignment. Make sure you have ALL bugs worked out of your StaticList module before beginning the implementation of this assignment.

Part I (Due Tuesday, October 19, 2010 by 5:00pm)

For Part I, you are to echo print all inputted information one line at a time using the above format. You are also to print the list lengths at each time step. No scheduling is to happen for Part I.

Here are the first few lines as a sample output:

```
     |            Planes Added             |       Runways       |    List  Lengths
Time | Takeoff  Landing (Fuel Remaining)   |  1   2   3  Crash  | Takeoff  Landing
---- | -------  -----------------------    | --- --- --- -----  | -------  -------
  1  |       2        3      (7   5   9)   |                     |    2        3
  2  |       2        3      (6   7   5)   |                     |    4        6
  3  |       2        3      (2   9   4)   |                     |    6        9
  4  |       3        2      (1   5   -)   |                     |    9       11
  5  |       2        2      (6   2   -)   |                     |   11       13
```

By 5:00pm on the date listed in Part I above, you are to submit a tarball called 05punetid.tar.gz that solves the Part I description. The tarball is to contain at least two projects: (a) StaticList and (b) AirportSimulation. In your AirportSimulation project, you will be referencing your StaticList functions. You need to make sure that you properly #include list.h when necessary from StaticList.

Part II (Due Wednesday, October 27, 2010 by 9:15am)

You are to complete the remainder of the assignment to perform the above task described.

Let’s go through the following Makefile example:

```
CC=gcc
CFLAGS=-g -Wall

all: airportdriver

airportdriver: Binaries/airportdriver.o Binaries/airport.o \
               ../StaticList/Binaries/list.o
       ${CC} ${CFLAGS} -o airportdriver Binaries/airportdriver.o  \
                 Binaries/airport.o ../StaticList/Binaries/list.o

Binaries/airportdriver.o: Sources/airportdriver.c  Headers/airport.h \
              ../StaticList/Headers/list.h
       ${CC} ${CFLAGS} -o Binaries/airportdriver.o -c Sources/airportdriver.c

Binaries/airport.o: Sources/airport.c Headers/airport.h \
              ../StaticList/Headers/list.h
       ${CC} ${CFLAGS} -o Binaries/airport.o  -c Sources/airport.c

clean:
       rm *.o airportdriver Binaries/*.o

valgrind:
       valgrind -v --leak-check=yes ./airportdriver
```

As always,

1. You are to break up your program into appropriate .h/.c files and on the day the assignment is due, turn in a colored hard copy of each .h/.c combination (fully documented).
2. Your code is to be written in C using Eclipse 3.6. Programs written in other environments will not be graded. Submit a tarball called **05punetid.tar.gz** using the submit script on zeus. Make sure to completely test your solution on zeus before submitting your final solution. This is a more complicated tarball and the tarball must extract and work correctly on zeus.
3. Test your modules one function at a time. This will lessen your level of frustration greatly. Start by creating a project called AirportSimulation. Then create your folders and a proper Makefile. Put in some simple stub functions and get your AirportSimulation project to compile while referencing a few simple list functions. This is not trivial!!!!
4. You are to use the coding guidelines from V6.0 of the coding standards.

Goals for this assignment:

1. Break your program up into well thought out modules.

2. Use well thought out functions in solving this problem. Don't break code out later into a function.

3. Code and test your program one function at a time.

4. Reuse your StaticList code

5. Write efficient/clean code

6. Use all of the tools in a more advanced ways including makefiles, the debugger, …

7. I will put out the datafile airport.txt at some point, but just use a small datafile such as:

```
2 3 7 5 9
2 3 6 7 5
2 3 2 9 4
3 2 1 5 0
2 2 6 2 0
```

THIS ASSIGNMENT IS HARD. START TODAY!!!!!