Airport

Date assigned: Friday, October 21, 2011
Date due: Monday, October 31, 2011, 11:59 pm
Points: 60

BONUS POINTS
by Oct 30 at 11:59pm +3 pts
by Oct 29 at 11:59pm +6 pts
by Oct 28 at 11:59pm +9 pts

You must receive **50** points or more to earn any bonus points.

For this assignment, you are to use your Priority Queue and Queue data structures to implement an airport simulator. You must simulate runway usage at an airport by determining which planes take off or land on each runway. Each plane that takes off flies to another airport and each plane that lands has taken off from a separate airport.

No new header or source files are provided for this assignment. The Q_DATATYPE structure is sufficient for this project.

You will need to implement an Airport module (airport.h, airport.c) that provides all the necessary functionality of the airport and an aiportDriver (airportDriver.c) that runs your airport simulation. You may also include a driver that thoroughly tests the functionality of aiport.c (airportTestDriver.c). I strongly encourage the creation of the Test Driver.

1. Your code is to be written in C using Eclipse 3.7. Programs written in other environments will not be graded. Create an Eclipse project named cs300_staticAirport_PUNetID. This project should contain the directories: src, include, and bin. The driver, airportDriver, should be created at the top level of the project, not in the bin directory.

2. The Makefile must contain the necessary targets to build the airportDriver as well as a clean and dist target similar to the identically named targets in your Stack assignment. There is no testMe target required for this assignment. Typing **`make`** on the command line should build airportDriver.

3. Submit a file called cs300_staticAirport_PUNetID.tar.gz into the CS300 Drop Box by 11:59pm on the day in which the assignment is due. This file must include your Airport, your Queue, your Priority Queue, and your Static List projects.

4. Submit a color, double sided, stapled packet of code by that same deadline. The packet should be in the following order:
    - Makefile
    - AirportDriver.c (.h then .c if you have both, otherwise just .c)
    - Airport.c (.h then .c if you have both)
    - Any extra .h/.c pairs you have. (do not include any code from previous projects)
    - Do NOT print airportTestDriver.c if you write it.

5. Test one function at a time. This will lessen your level of frustration greatly.

6. You are to use the coding guidelines from V6.0 of the coding standards.

7. The only changes to StaticList, pQueue, and Queue you can make are to fix bugs.

8. You must insert your Airport project into your Subversion repository.

Goals for this assignment:

1. Reuse your Static List, Priority Queue, and Queue project.

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

3. Write efficient/clean code

4. Use the debugger to effectively develop a correct solution

5. Thoroughly test your code.

Airport Simulation

The airport simulator is a turn-based simulator. Within each turn a number of events, specified below, occur. A clock (an integer) is kept to track the number of turns. Each turn takes one clock tick. Your simulator needs to determine which planes land and take off at each turn.

A plane may be sitting on the ground waiting to take off or a plane may be in the air waiting to land. The planes in the air have a non-negative integer amount of fuel. During each turn the fuel of each in-air plane is reduced by one. Once a plane in the air reaches zero fuel that plane must land before the next turn or that plane will crash. A priority queue, using fuel as priority, must be used to track planes waiting to land. This allows planes to land in priority order (zero fuel has highest priority, 1 unit has the next priority, etc.).

A non-priority queue must be used to track planes waiting to take off. Planes take off in the order in which they entered the system (FIFO). Both queues must store the clock tick (an integer) in which the given plane enters the system.

The airport has three runways. During each turn, each runway may either land exactly one plane or allow exactly one plane to take off. A runway may not both land and launch a plane in the same turn. A runway may also sit idle for a turn if no plane needs to land or take off.

The clock starts at 1.

A turn includes the following events in the following order:

1. Read a line of data from the file "data/airport.txt". Each line describes airplanes that are joining the take off queue and airplanes have arrived and need to land. Further, the amount of fuel on board for each newly arrived "need to land" plane is provided. Each plane arrives with a positive, non-zero integer amount of fuel. No fuel is assigned to planes that need to take off. At most, 3 planes may join the take off queue and an additional 3 planes may arrive needing to land (for a total of 6 new planes in the system per turn). It is also possible that zero planes enter the system at a given turn.
2. Enter the new planes into their appropriate data structures.
3. Decrement each "need to land" plane's fuel by 1 (including those that just arrived).

4. Those planes that "need to land" with a fuel value of zero must be assigned runways for landing. When all three runways are full any remaining planes in the air with zero fuel crash.
5. If step 4 did not use all three runways, the remaining runways are used. Service (land or take off) the plane at the head of the larger queue and remove that plane from its queue. If the queues are the same size, land a plane. Repeat step 5 until all runways are used or both queues are empty.
6. Print the results for the events of this turn.
7. Increment the clock by 1.
8. Return to step 1. Stop the simulation when both the file is exhausted and both queues are empty.
9. Print the summary statistics.

Airport Output

You must output the following table to the screen. Before the first clock tick and after each 20th clock tick reprint the table header (after clock tick 20, 40, 60, etc). The first line of numbers 012345... is for your reference and is not to be printed (Note that as with C arrays, the list of numbers starts with zero). Your output must look exactly as below, down to the spaces. There are 0 spaces after the final digit on each line in the table. There are no tabs. Each digit is right aligned. The data files I will run will not produce any digits that overflow too far to the left to disrupt the formatting (at most we will have 9999 clock ticks, for example). The summary statistics are printed with %g or as integers as appropriate. The data file that produced this output is given at the end of the document.

```
01234567890123456789012345678901234567890123456789012345678901234567890123456789
     |           Planes Added            |      Runways      |    List  Lengths
Time | Takeoff  Landing (Fuel Remaining) |  1   2   3  Crash | Takeoff  Landing
---- | -------  ------------------------ | --- --- --- ----- | -------  -------
   1 |       3        3 |    1    1    1 |  E   E   E      0 |       3        0
   2 |       3        3 |    2    2    2 |  T   T   T      0 |       3        3
   3 |       0        0 |    -    -    - |  E   E   E      0 |       3        0
   4 |       2        3 |    7    5    9 |  T   T   L      0 |       3        2
   5 |       2        3 |    6    7    5 |  L   T   L      0 |       4        3
   6 |       2        3 |    2    9    4 |  L   T   L      0 |       5        4
   7 |       3        2 |    1    5    - |  E   T   T      0 |       6        5
   8 |       2        2 |    6    2    - |  T   L   T      0 |       6        6
   9 |       1        0 |    -    -    - |  T   L   T      0 |       5        5
  10 |       0        0 |    -    -    - |  L   T   L      0 |       4        3
  11 |       0        0 |    -    -    - |  T   L   T      0 |       2        2
  12 |       0        0 |    -    -    - |  L   T   L      0 |       1        0
  13 |       0        0 |    -    -    - |  T   -   -      0 |       0        0

Average takeoff waiting time: 2.5
Average landing waiting time: 1.73684
Average flying time remaining on landing: 1.31579
Number of planes landing with zero fuel: 7
Number of crashes: 0
```

Key:
A – in the Fuel Remaining columns means there was no plane added to the system in that position. Line 3 above shows zero planes being added to the landing queue and line 7 shows only two planes being added to the landing queue. Therefore, on line 7 there are two digits in the Fuel Remaining columns followed by a -. All dashes must be in the furthest right column as possible.

Runways are marked as L, T, E, or -. A dash means the runway was unused. T means a plane used that runway to take off. E means a plane used that runway to land and the plane had zero fuel remaining (emergency landing). L means a plane used that runway to land and the fuel remaining was greater than zero.

Summary Statistics
You must track the necessary data to produce the above summary statistics. Average landing waiting time, average flying time remaining on landing (amount of fuel remaining), and number of planes landing with zero fuel each include those planes that crash.

Data File
The data file is guaranteed to not be corrupt or invalid. Zeros in various spots are valid. Fuel values may be up to four digits. Airplane crashes do not mean your simulation is not working. Some data files I provide may contain airplane crashes. If your airport simulator crashes that is an entirely different story.

Each **line** in the data file contains the following integers, separated by a single space, in this order:

Number Of New Planes That Want To Takeoff
Number Of New Planes That Want To Land
The amount of fuel on board for new landing plane 1
The amount of fuel on board for new landing plane 2
The amount of fuel on board for new landing plane 3

The amount of fuel is zero if there is no plane in that position.

```
3 3 1 1 1
3 3 2 2 2
0 0 0 0 0
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
1 0 0 0 0
```

This project is extremely difficult. Relying on three previously constructed projects can complicate your life as bugs in other projects are found and fixed. **Start early!**

For scale, I wrote about 800 (non-comment) lines of code for this project (airport.c, airportDriver.c).

Your StaticList driver, PriorityQueue driver, and Queue driver must still compile and run.