CS 2123
Summer 2014

Homework 3

Assigned: 6/16/2014
Due: 6/23/2014 11:59 pm

1. Implementing a Priority Queue as a Linear Linked List (50 pts)
Your program should read the included data file which contains a mixture of INSERTs and REMOVEs. Insert the provided ints into your queue and remove the smallest int each time a REMOVE is read from the file. I want you to maintain an ordered list, so your **pqinsert** function will need to do the heavy lifting and your **pqmindelete** only needs to remove the node at the front of the list. You should only have to track the front of the queue since it is an ordered list (i.e., you don't need to keep track of the rear of the queue).

The format of the data file is:
<COMMAND> <int>
or <COMMAND>

where COMMAND is either:
INSERT followed by a space and an int (all on the same line)
or REMOVE

For example:
```
INSERT 5
INSERT 3
INSERT 7
REMOVE
INSERT 1
```

would insert a 5 and then a 3 and then a 7 into your queue. Remember that I want you to implement an ORDERED linked list, so the 3 would be inserted at the front of the list. Remove would delete the smallest int (i.e., the one in the front) from the queue and then a 1 would be inserted.

For program output, I want to see how many nodes must be examined each time insert is called. I also want to see the contents of the queue on a separate line each time insert and remove are called. Lastly, add the # of checks for each insert to a global counter and print that on a separate line at the end of your program. Using the above example, your program's output would print something like:
```
Assignment 3 Problem 1 by El Professor
Insert into queue. Checked 0 nodes.
Queue contains: 5
Insert into queue. Checked 1 node.
```

```
Queue contains: 3 5
Insert into queue. Checked 2 nodes.
Queue contains: 3 5 7
Queue contains: 5 7
Insert into queue. Checked 1 node.
Queue contains: 1 5 7
Total checks: 4
```

2. Doubly Linked Lists (50 pts)
Copy your program in Problem 1 to a separate folder and change your Linear Linked List priority queue implementation to use doubly linked lists. You will need to change your node structure and some or all of your queue functions. You probably will not need to change your main (let me rephrase that: you should not need to change your main).

You will need to keep track of both sides of your priority queue: the min side (or front) and the max side (or rear). Remember, it is an ordered queue so the smallest int is the leftmost node (or the front or min) and the largest int is the rightmost node (or the rear or max).

Enhance your pqinsert function to start from EITHER the front or the rear when determining where to insert the new integer node. A simple way to do this is if the new integers value is > the max value / 2, then start your search from the rear, otherwise start your search from the front (in the same way as Problem 1).

Your program will produce the same sort of output as Problem 1. The queue contents will remain the same, but the number of checks should change.

3. Evaluation (10 pts)
Which implementation seems to be more efficient with respect to insertions? Roughly how much more efficient is the implementation? You don’t have to write much, but please write in complete sentences.

Deliverables:
Problems 1 and 2 should be submitted as SEPARATE C source code files. I need all of the C source that you produce for this assignment problem so that I may compile and execute it. Problem 3 should be a separate document (a text file TXT or RTF, or a PDF). Archive everything in a single zip file before submitting it to Blackboard.