lab 11: Queues

Instructions: Implement a Queue with whatever data structure you desire. Remember you may use code from a previous lab (*hint* *hint*).

In your lab finish the implementation of a Queue:

```
1
2  #ifndef QUEUE_H
3  #define QUEUE_H
4
5  template<class T>
6  class Queue {
7      private:
8          /* Class to implement.*/
9      public:
10         /* Empty constructor shall create an empty Queue! */
11         Queue();
12         /* Do a deep copy of queue into the this.
13          * Note: This one uses a reference to a Queue!
14          */
15         Queue(const Queue<T> &queue);
16         /* Deconstructor shall free up memory */
17         ~Queue();
18         /* Return the current length (number of items) in the queue */
19         int getLength() const;
20         /* Returns true if the queue is empty. */
21         bool isEmpty() const;
22         /* Print out the Queue */
23         void print() const;
24         /* Pushes the val to the end of the queue. */
25         bool push(const T &val);
26         /* returns the first element from the queue. */
27         T& first();
28         /* Removes the first element from the queue. */
29         void pop();
30         /* Returns if the two queues contain the same elements in the
31          * same order.
32          */
33         bool operator==(const Queue<T> &queue) const;
34   /* Add a value to the queue with respect to priority.
35    * For this function a lower number is a higher priority.
36    * EX: If the Queue is of integers and is { 5, 10, 15} and I add 7,
37    * the new queue is { 5, 7, 10, 15}.
38    */
39   void addWithPriority(const T& val);
40   /* Return the length of the shortest path, but with warps. The map is
```

```
    * a 2D array with each cell having the following property
    * 0: Cell is open (passable)
    * -1: Cell is a wall (impassable)
    * > 0: Cell is a warp.
    * A warp is a value > 999999 and warps to the position
    * (<first three digits, second three digits).
    * Hence 1000 warps to (1, 0), and 203109 warps to (203, 109)
    * To get the x value: <cell>/1000
    * To get the y value: <cell>%1000
    */
   int getShortestPathWithWarps(int **map, int width, int length, int sx, int sy,
                int ex, int ey);
};

#include "queue.cpp"

#endif
```

STL:
You may use vector, string, and list from the STL.

Write some test cases:
Create some test cases, using cxxtestgen, that you believe would cover all aspects of your code.

Memory Management:
Now that are using new, we must ensure that there is a corresponding delete to free the memory. Ensure there are no memory leaks in your code! Please run Valgrind on your tests to ensure no memory leaks!

How to turn in:
Turn in via GitHub. Ensure the file(s) are in your directory and then:

- $ git add <files>
- $ git commit
- $ git push

Due Date: Febrary 20, 2019 2359

Teamwork: No teamwork, your work must be your own.