lab 20 Priority Queues

Instructions: In this lab implement a priority queue using something better than O(n) for add and remove.

Implement the following interface:

```cpp
#ifndef PRIORITY_QUEUE_H
#define PRIORITY_QUEUE_H

template<class T>
class PriorityQueue {
    private:
        /* Class to implement.*/
    public:
        /* Empty constructor shall create an empty PriorityQueue! */
        PriorityQueue();

        /* Do a deep copy of queue into the this.
         * Note: This one uses a reference to a PriorityQueue!
         */
        PriorityQueue(const PriorityQueue<T> &pq);

        /* Deconstructor shall free up memory */
        ~PriorityQueue();

        /* Return the current length (number of items) in the queue */
        int getLength() const;

        /* Returns true if the queue is empty. */
        bool isEmpty() const;

        /* Print out the PriorityQueue */
        void print() const;

        /* Pushes the val to the top of the queue. */
        bool push(const T &val);

        /* Removes and returns the top element from the queue. */
        T pop();

        /* Returns if the two lists contain the same elements in the
         * same order.
         */
        bool operator==(const PriorityQueue<T> &pq) const;
};

```

```
#include "priorityqueue.cpp"

#endif
```

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: November 15, 2017 2359

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