Assignment 3: Priority Queues (small)

Out: 25th February 2014 **Due: 3rd March 2014 at 11:55pm**

Late due (20% penalty): 7th March 2014 at 11:59pm

In this assignment you will implement a template class for a priority queue using a binary **min-heap** data structure. The priority queue will store objects of any kind (you can assume that it is possible to compare two objects of that class using the "<" operator), each of which has a floating-point priority associated with it.

1. insert(T obj, float priority): Inserts a new object with the given priority.
2. T front(): Returns but does not remove the object with the **lowest** priority (by number) currently in the queue.
3. T pop(): Removes and returns the object with the **lowest** priority (by number) currently in the queue.
4. bool isEmpty(): Returns true if the priority queue is empty, false otherwise.
5. changePriority(T obj,float new_priority): Changes the priority of an existing object to the new priority. This function does not do anything if the provided object does not already exist in the priority queue.
6. remove(T obj): Removes a particular object from the priority queue. You can do this by first changing its priority to be the greatest in the queue, and then popping it off.

In order to implement the last two functions, you will have to append your priority queue with another data structure to locate a given object within the binary heap efficiently. Your priority queue should use an STL map for this purpose. The key of this map should be whatever type of object your priority queue stores, and its value should be an integer pointer. This pointer points to a single integer that stores the index of this object inside the binary heap's array. Similarly every entry in the binary heap, in addition to storing the object and its priority, should also store an integer pointer. **For an object, its entry in the binary heap and the map should both point to the same integer location.**

With this data structure in place, you must make sure all the operations above that change the heap or the map create/maintain/delete the integer pointer appropriately. For example, every time an object moves within the heap, its location in the heap changes and the map's entry should now be able to get to its new location correctly.

Finally, write a main function that tests your priority queue appropriately! By "appropriately" I mean evidence that you tried to insert more than 3 things in the queue, tested whether your queue works correctly with a random order of operations and each operation was tested.

What to submit

Please submit all .cpp files and .h files along with a Makefile in a single zipped file on reggienet.