HOMEWORK 7

100 points

DUE DATE: November 10th 11:59pm.

Warning: For any homework assignment that contains a programming segment please read the following very carefully.

Your code must compile and run on Black server. If your code does not work on Black server as submitted the grade for that problem is 0. Always test your code on Black, even if it is incomplete, make sure to get your code to compile and run on our Servers.

This homework contains 1 problem.

With your requirements document you are given:

1. An incomplete MaxHeap.h
2. main.cpp
3. Makefile
4. numbers.txt and numbers_small.txt

Please do not modify main unless it is specified in the problem

Problem 1

Your assignment is to build a max heap.
Max Heap has been discussed in class, please refer to lecture 11 notes on more guidance for a definition of Max Heap.
You need to support the following abilities. When you are implementing these methods listed below make sure to supply documentation for each method with pre and post conditions (see guidance additional guidance for pre post documentation at the end of this requirements document for a reference, you can also use the Queue lecture notes for reference):
Following is the list of required methods for your **MaxHeap.h** class.

Below are the listed methods that are required to be implemented:

Public methods:

- bool isEmpty(): It **returns** true if the heap is empty.
- const T getItem (int index): It returns item at particular index.
- void display(): It prints the heap in level order (breadth first traversal). It should throw an underflow exception if empty.

For example: Let's say we built our max heap and we invoked display method to see our values, the method should simply print one value at a time, see example below:

95
81
77
61
74
57
48

- int getHeight() : It **returns** the height of the heap.
- int getNumberOfNodes(): Returns the number of nodes in the heap
- void clear(): Removes all data from this heap.
- T peekTop(): Gets the data that is in the root (top) of this heap. For a maxheap the data is the largest value in the heap. Throws HeapUnderflow exception if empty.
- void add(const T& newData): Adds a new data item to this heap. This method should double the current size if needed before attempting to add.
- void remove(): Removes the data that is in the root (top) of a non empty heap, and ensures that it is still a max heap after removal.
- void increaseKey(int p, T positiveValue): This method increases the value of an item at position p in the heap by a positive amount and ensures that it is a max heap after this value update.
- void decreaseKey(int p, T positiveValue)This method decreases the value of an item at position p in the heap by a positive amount and ensures that it is a max heap after value update.
- myMaxHeap(vector<T> &input): This is the constructor of heap which builds a transform vector to a heap.

Private methods:

- void percolateDown(int hole) : This method will percolate down on heap to keep the max heap order. See Binary Heap lecture notes (Lecture11) for more guidance.
- Void percolateUp(int hole): This method will percolate up the heap to keep the max heap order.
- buildHeap(): This method reorganizes the vector to a heap. You should use the percolate down to build your heap.

Exceptions:

HeapOverflow and HeapUnderflow exception classes should be provided within MaxHeap.h (similar to earlier project with Stacks)

Remember all codes written in .h files must compile and run with our main method on Black Server.

Homework 6 Deliverables:

The following files must be submitted via Handin no later than November 10th 2016 by 11:59pm

1. **MaxHeap.h**

DOCUMENTING YOUR CODE WITH PRE AND POST CONDITIONS
BOTH DOCUMENTATION STYLES ARE ACCEPTABLE.PICK ONE!

EXAMPLE 1 Documentation is done as a header to a function.

```
109     /**
110      * @pre   : Heap must remain as max heap
111      * @param : x
112      * @post  :Adds an item to a max heap
113      * @return: None
114      */
115     void insert(const T &x) {...18 lines };
133
134
```

EXAMPLE 2: Documentation is done during listing of public methods in a class.

```
30      void Dequeue(ItemType& item);
31      // Function: Removes front item from the queue and returns it in item.
32      // Post: If (queue is empty) EmptyQueue exception is thrown
33      //       and item is undefined
34      //       else front element has been removed from queue and
35      //       item is a copy of removed element.
```