Project 4:
Min Max Heaps
Due: Check due date on BlackBoard

Introduction:
In this project, you will implement the min-max heap data structure described in Homework 4.

Assignment:
Your assignment is to implement the min-max heap data structure. Your implementation **must** be templated. **A Driver.cpp will be provided for you to test with, however it should be expanded upon.** The driver takes in **one** command line argument that will be a text file containing the data that will be inserted into the min-max heap. You **will** be tested on more data types than just integers, so be sure to test for this. (Similar to project 1.)

As stated in the homework, a min-max heap is a data structure that supports both deleteMin and deleteMax in $O(\log N)$ time per operation. The structure is identical to a binary heap, but the heap-order property is that for any node X:

at even depth:
the element stored at X is smaller than or equal to the parent but larger than or equal to the grandparent

at odd depth:
the element stored at X is larger than or equal to the parent but smaller than or equal to the grandparent

Classes:
The classes you must implement for this project are:
- MMHeap
- MyException

MMHeap.h/.cpp:
The MMHeap class will be your implementation of the templated min-max heap. This class must contain the following methods:
- int size() - returns the number of items in the min-max heap
- void dump() - prints out the contents of the min-max heap
- void insert(T x) – inserts a new value into the min-max heap
  - This method should run in O(log n) time. Insertion of duplicate values is allowed. Duplicate entries should appear multiple times in the MMHeap.
- T getMin() - returns the smallest value currently in the min-max heap
- T getMax() - returns the largest value currently in the min-max heap
  - getMin() and getMax() should run in constant time. If the heap is empty, these methods should throw a MyException exception.

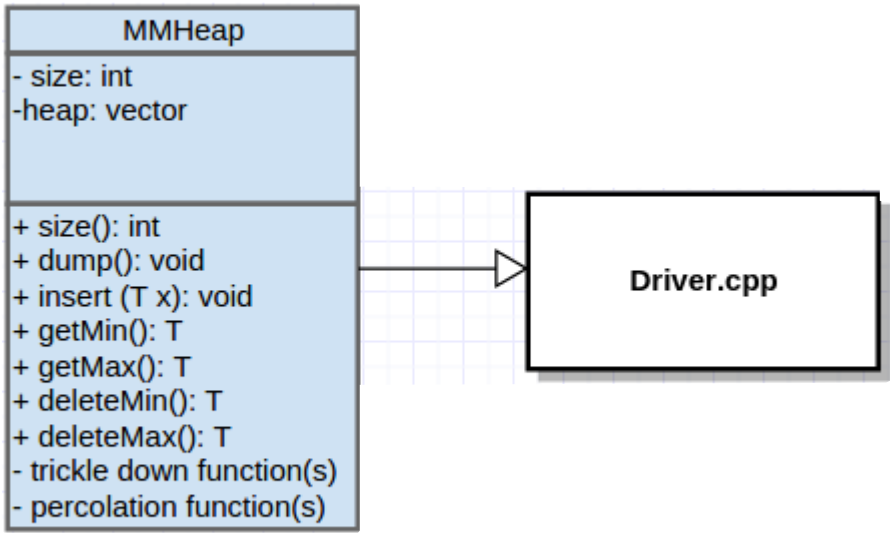
- T deleteMin() - removes **and** returns the smallest item
- T deleteMax() - removes **and** returns the largest item
  - deleteMin() and deleteMax() should run in O(log n) time.
- A solution for trickling down the heap.
- A solution for percolating up the heap.

Feel free to implement any additional helper functions you need to complete this part.

MyException.h:

This should be an exception that's thrown when data retrieval is attempted from a min-max heap when the heap is empty.

UML:



Implementation Notes:

Some recommendations and common pitfalls:

- Vectors in C++ do not like it when you pass the template data type directly to it. Consider another way of holding and accessing the data to put it into the vector.
- You need some way to keep track of whether the last level of your min-max heap is an odd level or an even level. Think through this. If you mess this part up, then your min-max heap will be messed up.
- In deleteMin(), when the new item trickles down and hits bottom, there are *two* reasons that may force it to percolate back up the max heap levels. Make sure you understand these two conditions. The situation is analogous for deleteMax().
- During a deleteMin(), there are 4 cases where you should stop the trickle down process:

1. You should stop when the key is actually in the right place. (It is smaller than the min items in the left and right subtrees.)
2. You should stop when the node has no children. (You actually hit bottom.)
3. You should stop when the node has 1 or 2 children, but no grandchildren.
4. You should stop when the node has 2 children, but only 1 or 2 grandchildren.

Note that it is impossible for a heap to have no right child, but the child has children. Also, if you have 3 or 4 grandchildren, the only reason you would have stopped trickling down is that the key is already in the right place. (Case 1)

You need to handle the four cases separately, because checking if you need to bounce up the maxheap levels has to be handled differently in each case. Also, Case 4 has subcases depending on whether the smallest item is the right child or one of the grandchildren. (An illustration of Case 4 can be found here here.)

Of course, the situation is analogous for deleteMax.

- There are special cases for deleteMax()when the min-max heap has only one or two items.
- You are probably better off making 4 separate methods that trickle down the min heap levels, trickle down the max heap levels, percolate up the min heap levels, and percolate up the max heap levels.

What to Submit:

Follow the course project submission procedures. You should copy over all of your C++ source code with .cpp/.h files under the src directory. You must also supply a Makefile build file. You should also

Make sure that your code is in the ~/cs341class/proj4/ directory and not in a subdirectory of ~/cs341class/proj4/. In particular, the following Unix commands should work.

Don't forget the Project Submission requirements shown online!! One hint, **after you submit**, if you type:

```
ls ~/cs341class/proj4/
```

and you see a bunch of .cpp and .h files, this is WRONG. You should see:

```
src
```

instead. The C++ programs must be in directories under the src directory. Your submissions will be compiled by a script. The script will go to your proj2 directory and run your makefile. This is required. You will be severely penalized if you do not follow the submission instructions.