Data Structures and Algorithms
Assignment 4

Due: 11/1/2021 in Canvas
(100 points)

Instructions

Submit your answers and code in Canvas. Do not submit the sample input files, submit only the `.hpp` and `.cpp` files (all of them, even if you do not modify the ones we are providing). Please put down your name in comments at the beginning of all the source code files. Include a brief README file explaining your code, especially if you implemented some of the suggestions for extra credits.

Question [100pt]

Implement MaxHeap and HeapSort.

In `assignment4.zip`, we are providing you with the following files:

```
public/
  demo.cpp     <-main method to try your implementation
  maxheap.cpp  <-implementation of maxheap (only file you need to modify)
  maxheap.hpp  <-header file
  inputs       <-folder with several sample inputs
```

•To compile, run

```
$ g++ -o heapsort demo.cpp maxheap.cpp
```

•To test your implementation with a sample input file, run

```
$ ./heapsort inputs/input.10.1
```

This will test a few methods (you need to check the maxheap after each step manually) and run heapsort (the code will check whether the output is sorted automatically).

•To test your implementation with all sample files with one command, run

```
time for f in inputs/input.10*; do echo $f; ./heapsort $f; done
```

The command `time` will let you know how long it took to run the code.

You may want to store the output in a file so that you can look at it carefully:

```
time for f in inputs/input.10*; do echo $f; ./heapsort $f; done > output
```

Grading

If you implement all the methods in maxheap.cpp properly (i.e., the demo works for any sample input), you will get 100. You also have a few opportunities to earn extra credits:

- Implement and test additional methods. For example, implement and test deleteMin.
- Optimize your code to run fast. The fastest submissions (top 5) will receive extra credits.