CSCI-1200 Data Structures — Fall 2020
Lab 13 — Priority Queues and Inheritance

In the first two checkpoints of today's lab, we'll implement *heap sort*, a classic $O(n \log n)$ sorting algorithm. Heap sort has better worst case performance than *quicksort* (which can suffer from poor pivot selection) and better memory usage than *merge sort* (which requires a scratch vector of size n).

Checkpoint 1

estimate: 15-25 minutes

Download this provided code:

http://www.cs.rpi.edu/academics/courses/fall20/csci1200/labs/13_priority_queues_inheritance/heapsort.cpp

Note that the provided code does not implement a templated Priority Queue / Binary Heap class, but instead works directly on input data stored in an STL vector.

Your first task is to complete the *heapify* function. This function takes in an STL `vector` of data and swaps the elements as little as necessary so that the data forms a proper binary heap. Refer to Section 25.13 from Tuesday's lecture notes. We'll heapify an unorganized vector by starting with the second to last level of the tree and call percolate down on every element in the second to last level. This will basically build lots of small 3-element heaps. Then we'll move to the third to last level of the tree and call percolate down on each of those elements. This will essentially join 2 3-element binary heaps with a new root node making many 7-element binary heaps. And so on. Until we get to the root of the entire structure.

Study the example tests in the provided code. Draw a picture of the data before and after the heapify process. First, work out on paper what the intermediate result (vector contents) should be after we call *heapify* for the two test input vectors.

To complete this checkpoint: Show a TA your diagrams and your debugged code for the heapify function.

Checkpoint 2

estimate: 15-25 minutes

Next, complete the implementation of the *heapsort* function. Test and debug your code. For your first draft you may use a scratch vector. Once that's debugged (if time allows) modify the code as needed to only use the input vector. *NOTE: You should not call* `push_back` *or* `pop_back` *on your vector!*

To complete this checkpoint and the entire lab: Show a TA your completed and debugged heap sort code.

Checkpoint 3

estimate: 30-60 minutes

Checkpoint 3 will be available at the start of Wednesday's lab.