CSCI-1200 Data Structures — Spring 2014
Lab 12 — Priority Queues and Binary Heaps

In this lab, you will use binary heaps to implement the priority queue container, as discussed in Lecture 21. Having these notes available while you are working on this lab will make it substantially easier.

Start by downloading the files, and then turn off all network connections:

http://www.cs.rpi.edu/academics/courses/spring14/ds/labs/12_priority_queues/priority_queue.h
http://www.cs.rpi.edu/academics/courses/spring14/ds/labs/12_priority_queues/test_pq.cpp

The code provided in these files is straightforward. `test_pq.cpp` is a driver and test program, while `priority_queue.h` is a skeleton implementation. Please take a careful look. You will complete the implementation and add to the main program in lab. In your implementation, be careful of subtracting 1 from an unsigned int whose value is 0; it is not -1!

Checkpoint 1

Implement and test the `push` (a.k.a. `insert`) and the `check_heap` functions. Recall that `push` depends on the `percolate_up` functionality. `check_heap`, which works either with the heap member variable or with a vector provided from the outside, determines if the vector is properly a heap, meaning that each value is less than or equal to the values of both of its children (if they exist).

To complete this checkpoint: Show a TA your debugged implementation and discuss the running time of both `insert` and `check_heap`.

Checkpoint 2

Implement and test the `pop` (a.k.a. `delete_min`) function and the constructor that builds a valid heap from a vector of values that is in no particular order. Both of these depend on proper implementation of the `percolate_down` function.

To complete this checkpoint: Show a TA these additions and the test output.