CIS 313 Lab 3: Priority Queues and Heaps

Due: November 13th, 2016 at 11:59pm

Overview

Use a binary Min-Heap to implement a Priority Queue.
Your priority queue class will be a wrapper for the functions in your heap class.
Your heap should be implemented using an array.
(ie) if a node is in array[k], then its left child is in array[k*2], and its right child is in array[k*2+1]
note: the root is in array[1]

Fill out all of the methods in the provided skeleton code.
You may add additional methods, but should NOT add additional public methods or public fields.
You also should not change the name of any of the classes or files.

In particular, you will implement the following functionality for your priority queue

Insert
Add priority p to the priority queue

findMin
return the highest priority from the priority queue

removeMin
remove and return the highest priority from the priority queue

isEmpty
print "Not Empty" if the priority queue is not empty, otherwise print "Empty"

print print out "Current Queue: " followed by each element separated by a comma. Do not add spaces between your elements.

Note: Your Priority Queue class should just be a wrapper for your heap class. Most of the work will be done in the heap class.

In particular, you will implement the following functionality for your heap

Insert
When adding a node to your heap, remember that for every node n, the value in n is less than or equal to the values of its children, but your heap must also maintain the correct shape.
(ie there is at most one node with one child, and that child is the left child.)
(Additionally, that child is farthest left possible.)

findMin
return the minimum value in the heap

extract
remove and return the minimum value in the heap

Extra Credit: Heapify
Given an array, create a heap in $O(n)$ time.

Grading
This assignment will be graded as follows:

Correctness 40%
Your program compiles without errors (including the submitted files NOT containing package names at the top. Delete the package name from the top of the files before you submit them): 10%
Your program runs without errors: 10%
Your program produces the correct output: 20%

Implementation 40%
Insert, removeMin, findMin, isEmpty, print all work correctly for the priority queue: 25%
Insert, extract, and findMin all work correctly for the heap: 15%

Documentation 20%
To earn points for implementation, your code must be clear enough for us to understand
Further, you may not use any data structures from the Java standard library or the C foreign function interface

Input
Input will be a number representing the number of following instructions followed by list of lines with different words specifying a task along with a number (if applicable).
Note: You should implement your priority queue with generics, but create a priority queue that takes in integers.

Sample input
11
insert 7
insert 10
insert 9
print
isEmpty
insert 8
findMin
insert 21
removeMin

print
removeMin

Sample output
Current Queue: 7,10,9
Not empty
7
7
Current Queue: 8,10,9,21
8