CIS 313, Intermediate Data Structures
Winter 2019

CIS 313 Lab 3

Due: Friday, March 1 at 4:55 pm

This lab involves implementing a Priority Queue using a Max-Heap

Overview

Use a binary Max-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 a list, L.
Recall, if a node is in $L[k]$, then its left child is in $L[2k+1]$, and its right child is in $L[2k+2]$

You should estimate an appropriate size for the queue based on the number of lines in the input file.
One method to do this could be to scan the file before instancing your priority queue, to read the number of insert statements or total lines.

Fill out all of the methods in the provided skeleton code.
You may add additional private 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

Maximum
Return the highest priority from the priority queue

extractMax
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. After you have printed all the elements in the Priority Queue make sure you print a new line character. This should represent the current structure of your heap.

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 greater than or equal to the values of its children, but your heap must also maintain the correct shape.
(i.e., there is at most one node with one child, and that child is the left child and that child is as far left as possible.)

Maximum
Return the maximum value in the heap

extractMax
Remove and return the maximum value in the heap

__Heapify
Used to maintain the structure of the heap after an element is added or removed from the heap

Input Description

The input will be a text file, for example *inSample.txt* below will be provided. Each line contains a different set of words specifying a task and a number (if applicable). You should create an empty heap (with root null), and then perform the prescribed sequence of actions on that tree.

```
insert 7
insert 10
insert 9
print
isEmpty
insert 8
maximum
insert 21
extractMax
print
extractMax
```

Output Description

Using the sample input above, your program should output:

```
Current Queue: 10,7,9
Not Empty
10
21
Current Queue: 10,8,9,7
10
```

Testing Protocol

We will test your program in the same fashion as previous labs. We strongly suggest you test your program in the following ways:

- While creating it
- With inSample.txt
- With your own examples and test-cases.

Grading

This assignment will be graded as follows:

Correctness 40%
Your program runs without errors: 20%
Your program produces the correct output: 20%

Implementation 40%
Insert, extractMax, Maximum, isEmpty, print all work correctly for the priority queue: 25%
Insert, extractMax, and Maximum 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 import any functions from the python library, or third party developers