CS 253 (Spring 2024)
Assignment 1: Heaps and Priority Queues

This project will review the ideas of Java's object oriented programming paradigms such as polymorphism and inheritance, as well as their limitations. In lecture we looked at an implementation of a heap based priority queue using an array to store the heap. In this homework project, you will build a binary tree structure to implement the heap in a file named `HeapTree.java`.

However, in Chapter 8, we have seen an implementation of a generic binary tree, in the file named `LinkedBinaryTree.java`. This class fully implements a binary tree ADT, but it is missing some of the functionality of a priority queue, such as swapping a parent and child within the tree.

We would not like to "reinvent the wheel" by rewriting a new binary tree structure from scratch, so you should use the principles of object oriented programming (such as inheritance and instantiation) to build your priority queue heap structure from the preexisting classes. At the same time, we do not want to copy the code from any of these files, since that would conflict with the notion of encapsulation. Your code would forever be tied to the implementation of the underlying binary tree class.

For this project, you should create two files.

- `HeapTree.java` This file should be a subclass of the `LinkedBinaryTree` class which adds the necessary tree operations that the priority queue will make use of, such as swapping a parent a child or replacing the root of the tree with another node.
- `TreePQ.java` This file should be a subclass of the `AbstractPriorityQueue` class, which uses the `HeapTree` to store the priority queue. (Hint: Whenever class A *uses* class B, it usually means that class A will instantiate class B as one of its data fields)

Once complete, the `TreePQ` class should be a functional heap based priority queue. You may add any `private` or `protected` helper methods as you would like.

While this project is not terribly complex, there are a few things that must be considered. Plan your structure and your strategies before you start coding in order to save yourself time.

You are responsible for testing to make sure that both your heap and your priority queue implementations can handle any input given to it.

Honor Code The assignment is governed by the College Honor Code and Departmental Policy. Please remember to have the following comment included at the top of the files.

```
/*
THIS CODE WAS MY OWN WORK, IT WAS WRITTEN WITHOUT CONSULTING ANY
SOURCES OUTSIDE OF THOSE APPROVED BY THE INSTRUCTOR. _Your_Name_Here_
*/
```

Submission:

Submit your completed `HeapTree.java` and `TreePQ.java` directly within GradeScope.

Grading for the program:

- If your program does not compile, you will get 0 points for the programming portion.
- The HeapTree class correctly extends the LinkedBinaryTree structure, adding the additional functionality. (40pts)
- Your heap based priority queue uses the tree structure of your HeapTree class and stores elements appropriately, maintaining the heap-order property and completeness property at all times. (40pts)
- Your structure correctly handles all possile cases, both expected and unexpected (10pts)
- Code clarity and style (10pts)