ITEC 324

Project 4

Heaps, Heap sort, and Recursion

For this project, you will be implementing a Heap data structure, and then using that heap data structure to facilitate a Heapsort of Strings.

Part (a) :

I've included a file called "Heap.java" that contains a class with the following methods:

- A constructor Heap() with no arguments that initializes an empty Heap object.
- A method findMax(), which returns the largest item in the Heap.
- A method extractMax(), which removes the largest item in the Heap and removes it.
- A method insert(T item), which accepts an item as input, and inserts the item into the Heap.
- A method isEmpty(), which returns true if the Heap is empty and false otherwise.
- A method size(), which returns the number of items in the Heap.
- A method height(), which returns the number of levels in the binary tree that represents the heap.

The Heap class accepts a type T as a generic parameter. T must implement the Comparable<T> interface. This means that, given two instances of type T called item1 and item2, you can test whether item1 > item2 by using the compareTo() method:

```
If (item1.compareTo(item2) > 0) then {
        // item1 is bigger!
}
```

The methods of this Heap class are not yet implemented. Implement these methods. Test your Heap implementation using unit tests to ensure that your Heap is implemented properly.

Additional design requirements:

- Your Heap class may not use arrays.
- Your Heap class may not include any imports. You may not use any built-in Java libraries or external libraries to implement your Heap.
- Your Heap class should use linked nodes. You must create your own Node class that represents a Node in the heap. Each Node should contain:
    - A reference to it's parent (the root's parent is null)
    - A reference to it's left and right children.
    - An instance of type T – the item stored at that node.
- Your implementation of the **Heap** may not use loops. You must use ***recursion*** *alone* to support all the heap operations.

    - Note: Your unit tests for the heap can use loops / imports. These restrictions are just for the Heap implementation itself.
- All operations on the Heap must take worst case $O(\log n)$ time.
- The Heap should always be a complete binary tree.
- You *may* add additional fields and helper methods within Heap.java as you see fit (make them private though).

Requirements for unit tests:

- Your unit tests do not need to use Junit, but it is recommended.
    - Alternatively, you can use a public static void main method, that calls other methods.
- At a minimum, you **must** have a test that:
    - Adds the numbers from 1 to 1_000_000 to your heap, in a random order.
        - You can use Collections.shuffle on an ArrayList<Integer> for this.
    - Repeat the following for each number i from 1_000_000 to 1:
        - Verify that findMax() returns i.
        - Verify that the height and size of the heap are correct.
        - Verify that the heap is not empty.
        - Remove the largest value i from the heap, and verify the returned result is i.
    - Verify the heap is empty after removing all the elements.
- You should add any additional tests that you see fit to ensure that your implementation is working correctly. (Consider interleaving inserts with extractMax()) calls.
- If you find that your code fails on the large test above, then using smaller tests will be necessary for debugging.

Part (b) : Heap sort

I've included a file "HeapSorter.java", with an empty main method. Write code for that main method that does the following:

- Reads lines of input from standard input (i.e., through System.in; for example by using Scanner.nextLine()) until an empty line is found or the end of input is reached.
- Outputs the lines, in sorted order, to standard output (i.e., using System.out).

Additional requirements:

- Your sorting code must use HeapSort, using the implementation of your Heap from part (a).
- Your code for part (b) may not use loops. You must use recursion to conduct all iteration (yes, that include input and output).
- You may not create any arrays or other data structures, besides your Heap itself.
    - You can store instances of *String*s, but not arrays of Strings or collections of Strings.
- Your should not include any imports, besides those necessary for reading lines of input (i.e., java.util.Scanner).
- Your should not use any external libraries.

- You *may* add additional fields and helper methods within HeapSorter.java as you wish (just not data structures).
- You should, of course, test your code to make sure it works properly, but you don't need to submit these tests.

Documentation and style requirements:

You must include, within your code, comments that explain your code's logic. Any non-trivial line of code should have a comment. I recommend you write out your logic completely in comments before you even start writing actual code (think of your comments as pseudocode). Then, when you actually write the code, you will have already worked through the logic.

The other purpose of this requirement is to help ensure that you actually understand the code your are writing!

Any fields and helper methods you add to the Heap.java class should be made *private.*

Grading rubric:

Requirement	**Points**
Recursive implementation of Heap methods	100
Unit tests for heap	20
Recursive HeapSorter main method implementation	50
Documentation and style: Including detailed explanation of code in comments	30
Total	200

Deliverables:

Submit your Heap.java file, your HeapSorter.java file, and your unit tests for your Heap to D2L by the posted deadline.