CS 3305: Data Structures

Assignment 6 – Heaps – 100 points

Note: If you re-upload the files, you must re-upload **ALL** files as the system keeps the most recent uploaded submission only. No zip files!

Note 2: Never hard-code test data in the test program, unless explicitly stated in the assignment. Always allow the user to enter the test data using menu option.

Note 3: Code documentation. In addition to the author header at the top of each file, please include a comment block before each method explaining the method and what it does. Add in-line comment to explain the code within the method.

The goal of this assignment is to reinforce the concepts of heap and heap sort, and to implement a priority queue as a heap tree.

Part 1 (20 points): Setup and test classes *Heap* and *HeapSort*.

For this assignment, we are using classes Heap.java (listing 23.10, page 878) and HeapSort.java (listing 23.10, page 879). The second file code is separated into 2 files: HeapSort.java and TestHeapSort.java. These three files are provided with the assignment.

Download these three files, compile and run class TestHeapSort.java. This class uses hard coded data for illustration purposes only. Notice the text data is of object type, not primitive type.

Extend the test class (TestHeapSort.java) to test (hard code) these 2 lists (character type and string type arrays):

```
list2 = {'w','f','A','X','T','Q','k','s','8','L','3','b','A','w','s','H','j','K','L'};

list3 = {"Data", "Structure", "Is", "Hard", "Computing", "Class", "To Pass"};
```

Compile and run your code and verify the correctness of outputs.

Part 2 (80 points): Implement Priority Queue using Heap structure.

Download and complete the implementation of class PQ_Heap.java, provided with this assignment. The only **data member** we need for this class is a **heap object** created from class Heap.java in part 1 above.

Hint: Create the **heap object** as private data member in class/file PQ_Heap.java; Then use the methods in class Heap.java to implement the priority queue methods outlined in class/file PQ_Heap.java by applying the heap methods on the heap object. In other words, implement class PQ_Heap using methods in class Heap.java. It is recommended that you implement class PQ_Heap.java as a **generic class**, similar to class Heap.java. In addition, to implement all priority queue methods outlined in class/file PQ_Heap.java, you may need to include additional methods in the generic class Heap.java.

Next, create a new test file named TestPQH.java to test class PQ_Heap.java. Use the following menu options as shown below (No menu, no points!). Start the program by displaying the menu first, then let the user interact with the program. Like you did in the previous assignment, **Force** the user to start with option 0 to select the data type of the priority queue content. Then allow the user to exercise other menu options on the selected queue type.

```
----------------MAIN MENU---------------
0. Enter Queue Type (integer or string)
1. Enqueue Element
2. Dequeue Element
3. Check is_Full
4. Check is_Empty
5. Print PQueue Size
6. Display Front Element
7. Print PQueue Elements
8. Exit program

Enter option number:
```

Like in the previous assignment, each menu option should display proper output with proper label. For example, option 5 would display something like this:

```
Testing method Print PQueue Size (Option 5)
PQueue size: 7
```

Follow the previous assignment sample outputs for other menu options for this assignment.

Note: The textbook author uses ArrayList to implement the heap, so we cannot have a static size for the ArrayList unless to impose a certain capacity for the heap. In order to implement options 3 and 4 above, we need to set the heap capacity to a certain size. For this assignment, add variable **`int CAPACITY = 100;`** in class Heap.java for this purpose. Also notice that in class Heap.java, method **`list.size()`** gives you the current size of the heap.

Always re-display the menu after an option (other than option 8) is fully exercised with blank lines before and after the menu. Handel special cases, such as an empty queue, and print proper messages to describe the queue status.

For option 7, print the PQ content as shown below. Notice that value 99 below is the parent node for values 66 (left child) and 44 (right child); value 66 is the parent node for values 33 (left child) and 22 (right child); Nodes 44, 33, and 22 have no child nodes on the heap.

```
Index 0:  99   66   44
Index 1:  66   33   22
Index 2:  44
Index 3:  33
Index 4:  22
```

Submission:

Do not forget to include the author header in each submitted file as shown, and do not forget to document your code as stated in note 3 above. **no author header or no proper documentation, no points!**

```
// Name:          <your name>
// Class:         CS3305/W03
// Term:          Spring 2025
// Instructor:    Prof. Wang
// Assignment:    #6
// IDE Name:      <your IDE name>
```

Please submit all .java files (total 5 files), named as indicated above, to the assignment submission folder in D2L by the due date posted in D2L (Heap, HeapSort, PQ_Heap, TestHeapSort, and TestPQH). Make sure that your code is running correctly right before you upload your files. **No zip files or late submissions are accepted**.