EECS 233 Homework 6

General requirements:

- Due at 11:00 PM on the posted due date.
- Include your name and network ID as a comment at the top of all of your programs.
- Create a typed document (.txt or .pdf) with answers to the questions.
- Upload your document and all .java files as a .zip file to Blackboard. Do not use other formats such as .rar.
- All work should be your own, as explained in the Academic Integrity policy from the syllabus.

Instructions: This assignment requires you to work with binary heaps using the program BinaryHeap.java that was discussed (and provided) in class.

1. In your report, draw the state of a binary heap after each of the following (in order a – n):
   a. Insert 9
   b. Insert 3
   c. Insert 2
   d. Insert 1
   e. Insert 6
   f. Insert 7
   g. Insert 5
   h. Insert 4
   i. Insert 8
   j. Remove minimum
   k. Remove minimum
   l. Remove minimum
   m. Remove minimum
   n. Remove minimum

2. Using the tree created in problem #1 for steps (a) through (i), not including removals, answer the following in your report. Note that this tree will have 9 nodes.

   a. What are the contents of the array? List them in order, starting with index #0 and ending with the maximum index for the array (not the last value in the heap). The maximum index for the array is the largest index for a full tree with the same number of levels.

   b. Give three (3) more unique insertion sequences that would generate the same tree. To be different, two or more values must be in a different order.

3. Revise BinaryHeap.java to be a maximum heap where all children are less than the parents. You may name it MaxHeap.java if you wish, but it is not required. Note that everything "Min" should now be "Max", including "findMin" and "deleteMin". You will be able to test your changes in the next problem.

4. Add the following methods to your new heap from problem #3 above. Demonstrate the methods after using the "insert" method to create a heap.

   a. Add a method that prints all of the heap values in order by array index in a single row.

   b. Add a method that prints each row of the heap tree on a separate line.

   Sample output using the following insertions: 9,3,2,1,6,7,5,4,8

```
Array:
9 8 7 6 3 2 5 1 4
Tree:
9
8 7
6 3 2 5
1 4
```

Tip (checking your tree): Though not required, you can add a method to BinaryHeap.java to print the tree as you are required to do for problem #4. You could use this to check your answer for problem #1.

Tip (different sequences): Remember that you must use the same values in a different order. To find more sequences that create the same tree, it is recommended that you think of a strategy for changing previous sequences. However, you may also find that you get lucky using random changes. Here's one that <u>won't</u> work: 1,2,3,4,5,6,7,8,9.

Tip (bubbling in a maximum heap): The bubbling (percolation) directions remain the same for insertions and deletions. Only the relationship between parents and children is different for the regular minimum heap used in problems #1 and #2.

Tip (printing a tree): There are various ways to print the tree, but the primary task is to determine when one row stops and the next row begins. One approach is to compute how many numbers should be printed in each row: 1, 2, 4, 8, etc. For fun (not for credit), you might try to print a more attractive tree with horizontal symmetry.

Rubric:

Item	**Points**
1: insertions/removals (14 steps)	15
2: array contents	15
2: three sequences (5 points each)	15
3: changing method names appropriately	5
3: logic changes	10
4: method to print array	15
4: method to print tree (basic printing)	15
4: method to print tree (separating rows)	10
Total	100