Name _____

Final Exam

IMPORTANT

This exam consists of **5 questions**, each worth **20 points**.
Please name your files: q1.java, q2.java, etc.

When you've finished, please zip the five files in a single **exam.zip** file and upload it to ReggieNet.

Question 1 (20 points)

Consider an array of integers a (int[] a). We would like to generate a 2D array (i.e., matrix) m (int[][] m) as follows:

- Initially, all element of the 2D array m are set to 0.
- Take a[0], the first element of the array a and assign it to m[0][0] (i.e., m[0][0] = a[0])
- for any subsequent element a[j], j > 0:

  If a[j-1] <= a[j], continue by inserting a[j] in m in the same direction (left to right or downwards)

  If a[j-1] > a[j], flip the direction of insertion, i.e., if you last inserted left to right, then now start inserting the elements of a in m downwards and, if you last inserted downwards, then now start inserting the elements of a in m left to right.

Example:

a

2	3	5	7	9	1	4	8	3	5	2	3	7	11	4	4	3

m

2	3	5	7	9	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	4	0	0	0	0
0	0	0	0	8	3	5	0	0
0	0	0	0	0	0	2	0	0
0	0	0	0	0	0	3	0	0
0	0	0	0	0	0	7	0	0
0	0	0	0	0	0	9	0	0
0	0	0	0	0	0	11	4	4
0	0	0	0	0	0	0	0	3

Write a java program that randomly populates an array a of 40 integers between 1 and 100 and then generates a 2D array m as explained above.

Your program must then print out both the original array a and the obtained 2D array m.

Question 2 (20 points)

Write a Java program that:

1. Randomly populates a **stack S** with 30 integers whose remainder of division by 10 is 3, 5 or 7. For example, the following are numbers that satisfy this:

   15, 13, 43, 57, 63, 87, 95, 5, 3, 7, 17, 53, 65, 73

2. Prints out the **content of the stack S**.

3. Uses the content of **stack S** to create **two other stack S5 and S7** such that, at the end:

   a. Stack S contains only the numbers whose remainder of division by 10 is 3
   b. Stack S5 contains only the numbers whose remainder of division by 10 is 5
   c. Stack S7 contains only the numbers whose remainder of division by 10 is 7

Important:

- throughout the process, **no number** should be removed (using pop()) from the original stack S **twice**.

- You should not use any additional temporary data structures (other stacks, arrays, etc.)

4. At the end, prints out the content of all three stacks: S, S5, and S7

Question 3 (20 points)

Write a program that has a **main()** function that calls a **recursive method repeatPrint(String s, ...)** that prints out the string s as shown in the example below:

```
repeatPrint("CHICAGO", ….);
```

```
            C
          C H
        C H I
      C H I C
    C H I C A
  C H I C A G
C H I C A G O
C H I C A G O
H I C A G O
I C A G O
C A G O
A G O
G O
O
```

Question 4 (20 points)

In class, we discussed a recursive **modified** version of the selection sort algorithm that we called **minmaxSelectionSort()**. Figure 1 shows a version of minmaxSelectionSort() that contains **three errors**.

```
static void swapElements(int[] tab, int i, int j)
   {
       int tmp = tab[j];
       tab[i] = tab[j];
       tab[j] = tmp;
   }


 static void minmaxRecursiveSelectionSort(int[] tab, int i, int j)
   {
       int min = 0;
       int max = tab.length - 1;

       if (i > j)
           return;

       for (int k = i + 1; k <= j; k++)
       {
           if (tab[k] > tab[max])
               max = k;
           if (tab[k] < tab[min])
               min = k;
       }

       swapElements(tab, i, min);
       swapElements(tab, j, max);
       minmaxRecursiveSelectionSort(tab, i + 1, j - 1);
   }
```

Figure 2.

1. Find and fix the three errors.

2. We would like to write a new method **reverseSort(.....)** that sorts its input in **descending order**. Write the method reverseSort() using ONLY the given two methods: minmaxSelectionSort() and swap()

   Note: the method reverseSort() does not have to be recursive.

Question 5 (20 points)

Let sumDigits(x) be the sum of the digits of integer x. For example: sumDigits(53) = 8, SumDigits(325) = 10.

1. Write a program that generates a sequence of random integers between 1 and 100 and builds a binary tree in a recursive way as follows:

- The root of the tree is the first integer generated

- To insert a new integer in the tree, you need to traverse the tree recursively. When you visit a node n (whose integer value is n.value) to insert a value x:

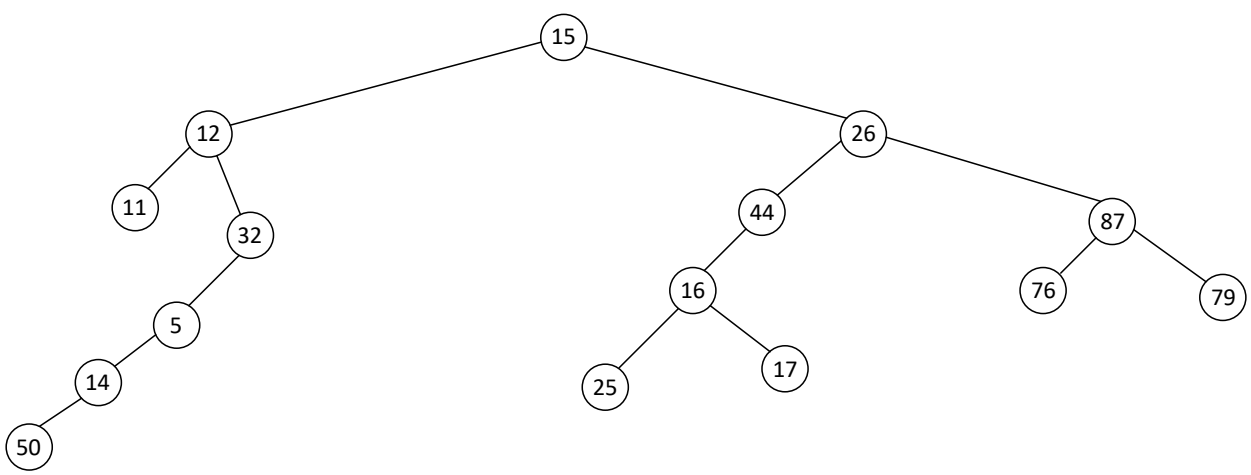
    i. If the difference sumDigits(x) <= sumDigits(n.value), insert x in the left subtree of node n.
    ii. If the difference sumDigits(x) > sumDigits(n.value), insert x in the right subtree of node n.

Example:

Input (randomly generated in this order)

15, 12, 26, 11, 32, 87, 44, 5, 14, 76, 79, 50, 16, 17, 25

Binary Tree:



2. Write a method **findMax(root)** that takes as a parameter the root of the tree **root** and prints out the **highest value in the tree** and the **level** of that node. For example, in our example, the output should be:

Max = 87, Level = 3