CISC 1115 Spring 2021 Final Exam Questions & Answers

1. (12 points) Write Java code to repeatedly print each single digit between 1 and 9 the number of times based upon its numeric value. Thus, you would get a triangle of the shape below where 1 prints once, 2 twice … and 9 prints 9 times.

```
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

2. (14 points) Write Java code to read strings of lower-case letters from the keyboard and count the number of vowels in each word. (vowels are a,e,i,o and u). When all strings have been read in, print the string that has the largest number of vowels (duplicates included) and how many vowels were in the string. When more than one string has the greatest number of vowels, print the first string found with that number. For example, with input of **she groceries yourselves here radio**, the correct answer would be: **groceries – 4 vowels**.

3. (14 points)
   a. Show how the code executes step-by step. Display (trace) the values of x and y as they change.

```
int x = 7;
int y = 3;
//print x and y values at this point
while (x+y < 13){
   x = x + 2;
   y = y - 1;
   //print x and y values at this point
   do {
     x--;
     y++;
     //print x and y values at this point
   } while (x>10);
 }
```

x									
y									

   b.

```
for (int i=12; i>9; i--){
   for (int j=i-2;j< 15;j=j+2)
   //print i and j values at this point
}
```

i										
j										

4. (11 points) Fill in the table below to trace the values of the arrays in main at various points.

```
int[] array1 = {10,20,30,40,50};
int[] array2 = {5,4,3,2,1};
int[] array3 = new int[5];

//print the array values at this point
doIt(array1, array2);
//print the array values at this point

array2 = array1;

//print the  values at this point
array3 = doIt(array1,array2);
//print the array values at this point

} //end of main
```

```
public static int[] doIt(int[] arr1, int[] arr2){
    int[] arr3 = new int[5];
    for (int i=0; i<arr1.length;i++)
        arr3[i] = arr1[i]+arr2[arr2.length-1-i];
    return arr3;
}
```

Values of the arrays below in main before first invocation of doIt	
array1	
array2	
array3	
Values of the arrays below in main after first invocation of doIt	
array1	
array2	
array3	
Values of the arrays below in main before second invocation of doIt	
array1	
array2	
Values of the arrays below in main after second invocation of doIt	
array1	
array2	
array3	

(5 or 6 13 points pick one)

5. You are playing a game that has two dice – one die has 6 sides numbered 1 through 6 and the second die has 8 sides numbered 1 through 8. A turn involves rolling both die together. Write Java code to play 1,000 turns and keep track of the sum of the two dice in each roll i.e, (2 through 14). When all 1,000 turns have been completed, print a formatted table showing the value of the sum of the two dice (2 through 14) and the number of times that value occurred.

6. Write Java code to do the following: Read from the keyboard an unknown number of values each of which is between 1 and 100. If a number entered is not between 1 and 100 (except for -1 which ends the input) print an error message. When a -1 is read in, stop the loop and print which of the numbers between 1 and 100 were <u>not</u> read in.

7. (6 points – 2 points each) *Perform the following conversions**. You must show how you computed your answer.***

   a. 1011111 (base 2) to base 10 Answer _____

   b. 246 (base 10) to base 2 Answer _____

   c. BA (base 16) to base 2 Answer _____

8. (30 points) Write a complete Java program with comments in main and in each method.

A city is divided into 100 neighborhoods, each with a unique name. Every three months, each neighborhood reports the prices of four houses sold; not all neighborhoods have sales to report every three months. A file contains the sales data in the format:

neighborhood price price price price

For example, Midtown 23055 10000 19000 32009 (Note: prices are in whole dollars)

Design a **Java class** with a **main** method that does the following:

1. Invokes method **readData** which reads the data from the input file, stores values into arrays and returns the number of records read in.
2. Invokes method **modifyData**, passing an array of double as the parameter. The method modifies the value in the array based on rules specified below.
3. Invokes method **sortArrays** to parallel sort the arrays of double and String. The method should be invoked only once.
4. In main, prints to a file (name of your choice) the neighborhood name and average of the three neighborhoods that have the highest average prices, in descending order (highest average price first), and the three neighborhoods that have the lowest average prices in ascending order (lowest average price first). The neighborhood should be left adjusted, the price right adjusted with two decimal places and the header row should be included. The output should be in the form:

```
Neighborhood   Avg Price
Eastside       108162.50
Greenfields     67576.06
Southside       60967.78

Neighborhood   Avg Price
Chelsea         47663.28
Westside        48830.93
Uptown          57200.00
```

Method Details:

I. **readData:**
   a. Receives an array of String and array of double
   b. Reads the neighborhood name as String and the four prices as integer from a file (name of your choice)
   c. Stores the neighborhood name in the array of String and the average of the 4 prices in the array of double.
   d. Returns the number of neighborhood records read in as an integer

II. **modifyData:**
   a. Receives an array of double as the parameter and an integer representing the number of records read in by **readData**
   b. Computes the overall average home price for all homes read in by **readData** (the average of the averages)
   c. For the number of records read in from the input file, increases the average price for each neighborhood by 10% if the average home price for that neighborhood is below the overall average home price homes sold and decreases the average price for each neighborhood by 15% if the average home price for that neighborhood is above the overall average.

III. **sortArrays:**
   a. Receives the arrays of double and String and an integer representing the number of records read in by **readData** as parameters
   b. Parallel sorts the arrays, only for the number of records read in by **readData**, with the primary sort on the array of double in descending order.

1. (A)

```
for (int i=0; i<10; i++){
 for (int j=0; j<=i-1;j++){
     System.out.print(i);
 }
 System.out.println();
```

(B)// One way

```
        for (char ch= 'A'; ch<='Z'; ch++){
          for (int i=0; i<=ch-'A';i++){
              System.out.print(ch);
          }
          System.out.println();
         }
// Another way
       char ch = 'A';
       for (int i = 0; i<26; i++){
        for (int j=0; j<=i;j++){
           System.out.print(ch);
         }
         System.out.println();
         ch++;
        }
   // and a third way
        int max=1;
        for (char chr = 'A';chr<='Z';chr++){
            for (int i=0;i<max;i++){
                System.out.print(chr);
            }
            System.out.println();
            max++;
        }
```

2.

```
   Scanner input = new Scanner(System.in);
 int maxVowels = 0;
 String maxStr="";

 System.out.print("Enter string,999 to end: ");
 String str = "";
 while (input.hasNext()){
     str = input.next();
     if (str.equals("999")) break;
     int count=0;
     for (int i=0;i<str.length();i++){
         if (str.charAt(i)=='a'|| str.charAt(i)=='e'||
             str.charAt(i)=='i'|| str.charAt(i)=='o'||
              str.charAt(i)=='u' )
             count++;
     }
     if (count > maxVowels){
         maxVowels = count;
         maxStr = str;
     }
     System.out.print("Enter string,999 to end: ");

 }
 System.out.println("String with max vowels is: "+ maxStr +" with "+maxVowels);
```

3.

x	7	9	8	10	9	11	10		
y	3	2	3	2	3	2	3		

i	12	12	12	11	11	11	10	10	10	10
j	10	12	14	9	11	13	8	10	12	14

4. **(A)**

Values of the arrays below in main before first invocation of doIt	
array1	**10 20 30 40 50**
array2	**5 4 3 2 1**
array3	**0 0 0 0 0**
Values of the arrays below in main after first invocation of doIt	
array1	**10 20 30 40 50**
array2	**5 4 3 2 1**
array3	**0 0 0 0 0**
Values of the arrays below in main before second invocation of doIt	
array1	**10 20 30 40 50**
array2	**10 20 30 40 50**
Values of the arrays below in main after second invocation of doIt	
array1	**10 20 30 40 50**
array2	**10 20 30 40 50**
array3	**60 60 60 60 60**

(B)

```
array1: 20 40 60 80 100
array2: 2 4 6 8 10
array3: 0 0 0 0 0

array1: 20 40 60 80 100
array2: 2 4 6 8 10
array3: 0 0 0 0 0

array2 = array1
array1: 20 40 60 80 100
array2: 20 40 60 80 100
array3: 0 0 0 0 0
array3 = doit(array1,array2)

array1: 20 40 60 80 100
array2: 20 40 60 80 100

array3: 120 120 120 120 120
```

5.

(A)

```
int [] pct = new int[15];
int sum =0;
int runs = 1000;
for (int i=0; i<runs; i++){
    int roll  = throwDie(1,6)+throwDie(1,8);
    pct[roll]++;
}
System.out.printf("%4s%6s%7s", "roll","count");
for (int i=2;i<15;i++){
    System.out.printf("\n%4d%6d%7.2f",i,pct[i]);
}
System.out.println();

// method - can use Random class
// not necessary to write a method
public static int throwDie(int start, int end){
    return (start+ (int)((Math.random()*(end-start+1))));
}
```

(B)

```
Scanner input = new Scanner(System.in);
boolean[] flag = new boolean[101];

System.out.println("Enter number between 1 and 100 or -1 to end: ");
int num = input.nextInt();

while (num != -1){
    if (num > 0 && num < 101)flag[num] = true;
    else System.out.println
      (num + " is invalid. Number must be between 1 and 100");

    System.out.println
       ("Enter number between 1 and 100 or -1 to end: ");
    num = input.nextInt();

}
System.out.println("These numbers were not read in:");
for (int i=1; i<101;i++){
    if (!flag[i])
        System.out.print(i+" ");
}
```

6. (A) a. 95 b. 11110110 c. 10111010

   (B) a. 93 b. 10111010 c. 11111001

7.

```
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws IOException {
        File fileOut = new File("C:\\Temp\\buildingOutput.txt");
        PrintWriter output = new PrintWriter(fileOut);

        String[] names = new String[100];
```

```
        double avg[]  = new double [100];
        int numRecs= readData(names, avg);
        modifyData(avg,numRecs);
        sortArray(avg, names,numRecs);

        // top 3, highest first
        output.printf("\n\n%12s%12s","Neighborhood","Avg Price");
        for (int i=0; i<3;i++){
          output.printf("\n%-12s%12.2f",names[i], avg[i]);

        }
        // lowest 3, smallest first
        output.printf("\n\n%12s%12s","Neighborhood","Avg Price");
        for (int i=numRecs-1; i>numRecs-4;i--){
          output.printf("\n%-12s%12.2f",names[i], avg[i]);
        }
        output.close();
    }
        public static int readData(String[] names, double[] avg)
                                                    throws IOException{
            File fileIn = new File("C:\\Temp\\buildingInput.txt");
            Scanner input = new Scanner(fileIn);
            int count=0;
            while (input.hasNext()){
                double average = 0;

                String name = input.next();
                int[] price = new int[4];

                for (int i=0; i<4;i++) {
                    price[i] = input.nextInt();
                    average = average + price[i];
                }

                names[count] = name;
                avg[count] = average/4.0;
                count++;
            }
            input.close();
            return count;

     }
        public static void modifyData(double[] avg, int num){
            double overallAvg = 0;
            for (int i=0; i<num;i++){
                overallAvg = overallAvg + avg[i];
            }
            overallAvg = overallAvg/num;

            for (int i=0; i<num;i++){
                if (avg[i]<overallAvg) avg[i] =  1.1*avg[i]; (B) = .85*avg[i]
                else avg[i] =  .85*avg[i];                    (B) 1.2*avg[i]
            }
        }

        public static void sortArray(double[] primary,
                String [] secondary, int k){
         for (int i = 0; i < k; i++)
         {
            for (int j =0; j < k-1-i; j++)
            {
                if ((primary[j]< primary[j+1]))  (B) >
                {
                    double smaller = primary[j];
                    primary[j] = primary[j+1];
                    primary[j+1] = smaller;
                    String small = secondary[j];
                    secondary[j] = secondary[j+1];
                    secondary[j+1] = small;
                }
            }
         }
    }
}
```