CISC 1115 Fall 2018 Final Exam

Question 1. (12 points – 4 points each)

Each of the parts a. through c. below is preceded by a comment indicating what the code should do. There is at least one problem with each section of code and it fails to do what was intended. Show how to modify or rewrite the code so that it does work as intended. If there are multiple problems, correct each one. Assume that all variables used have already been declared.

```
a. // INTENT: given an array arr of int values
   // set small equal to the smallest of the array values

        small = arr[0];
        for (int j = 0; j < arr.length-1; j++) //does not check entire array, checks one less
          if (small < arr[j])//does not work because we should be looking for arr[j] to be
smaller than the small we already have
              arr[j] = small; //assigns the value of small (arr[0]) to arr[j] if small is smaller
than arr[j]
```

should be:

```
        small = arr[0];
        for (int j = 0; j <= arr.length-1; j++)//either add = to original or remove -1
          if (small > arr[j])
              small = arr[j] ;
```

```
b. // INTENT: compute the average of all n values in the integer array arr;
   // compute the average as a double and store it in the double variable avg

        int sum = 0;
        int n = arr.length;
        for (int k = arr.length; k <= 0; k--) //k cannot be equal to length (out of bounds), k
can never be less than 0 (out of bounds)
             sum = sum + arr[k];
        double avg = sum / n; //integer division
```

should be:

```
        int sum = 0;
        int n = arr.length;
        for (int k = arr.length-1; k >= 0; k--)
             sum = sum + arr[k];
        double avg = (double) sum / n; //cast to double before division
```

```
c. //INTENT: the array b should have the values of array a in reverse order
       int [ ] a = {1,2,3,4,5};
       int [ ] b = new int[5];
       for (int i=0; i<=a.length; i++)
              b[i-1] = a[a.length-i];//index out of bounds: b[-1] = a[5] when i = 0
```

should be:

```
       int [ ] a = {1,2,3,4,5};
       int [ ] b = new int[5];
       for (int i=0; i<=a.length; i++)
              b[i] = a[a.length-1-i];
```

Question 2. (15 points)

What is printed by each of these pieces of code?

a.
```
int x = 5;
do {
        System.out.println(x);
        x = x + 3;
    } while( x < 10 );
    System.out.println(“now “+ x);
```

```
5
8
now 11
```

b.
```
for (int i = 3; i > 1; i--) {
    System.out.println(i + “ and " + (i * 10 – 1));
}
System.out.println("finished");
```

```
3 and 29

2 and 19

finished
```

c.
```
double z = 5.4321;
int j = 4;
while (j < z) {

    System.out.println(j +" is less than " + z);
    j++;
}
System.out.println ("it is " +j +" at the end");
```

```
4 is less than 5.4321
5 is less than 5.4321
it is 6 at the end
```

Question 3. (13 points)

What is printed by the following sections of code?

a.(5 points)
```
String source = "ID is 12-AB-1X ";
int pos = source.indexOf('-'); //8
int pos1 = source.indexOf(' ',pos+1); //14
String str1 = source.substring(pos+1,pos1);
System.out.println(str1);
```

```
AB-1X
```

b. (5 points)
```
source = "from Denver,CO.";
int pos2 = source.lastIndexOf(','); //11
int pos3 = source.lastIndexOf(' ',pos2); //4
String str2 = source.substring(pos3+1,pos2);
System.out.println(source.substring(pos3+1,pos2));
```

```
Denver
```

c. (3 points)
```
System.out.println(str1.compareTo(str2)>0);
```

```
false
```

Question 4. (6 pts)

Perform the following conversions :

a. 101010 (base 2) to base 10

Answer _____**42**_____

101010 >> 6 digits >> base 2^5 - 2^0

(1 * 2^5) + (0 * 2^4) + (1 * 2^3) + (0 * 2^2) + (1 * 2^1) + (0 * 2^0)

32 + 0 + 8 + 0 + 2 + 0 = 42

b. CA (base 16) to base 10

Answer ______**202**______

CA >> 2 digits >> 16^1 - 16^0

C = 12, A=10

(12 * 16^1) + (10 * 16^0)

192 + 10 = 202

c. 10101011 (base 2) to base 16

Answer _______**AB**_______

10101011

1010 1011

1010 = A

1011 = B

d. 50 (base 10) to base 2

Answer ______**110010**________

```
50 / 2 = 25        remainder = 0
25 / 2 = 12        remainder = 1
12 / 2 = 6         remainder = 0
6 / 2 = 3          remainder = 0
3 / 2 = 1          remainder = 1
1 / 2 = 0          remainder = 1
```

1 1 0 0 1 0

e. F8 (base 16) to base 2

Answer ______**11111000**______

F = 1111

8 = 1000

f. Perform the following addition of two binary (base 2) numbers. Your answer should be a number in binary.

101 + 110 = ? //5 + 6 = 11

Answer ______**1011**_______

Question 5. (12 points) //Good for showing that arrays are pass by reference, changes made in methods stick

What is printed by the following code found in main of a Java program with the methods below?

```
int[] arr = {1,2,3,4,5};

doOne(arr[0]);
System.out.println(arr[0]);
```

prints out: 1 //individual int pass by value, changes in method NOT in main unless returned

```
doTwo(arr,1);
System.out.println(arr[1]);
```

prints out: 40 //array is pass by reference, changes made in method DO stick so that it is changed in the entire program, arr now looks like: {1, 40, 3, 4, 5}

```
doThree(arr);
System.out.println(arr[2]);
```

prints out: 15 //array is pass by reference, once again changes made in method DO stick so that it is changed in the entire program, arr now looks like: {5, 10, 15, 20, 25} IF CALLED BEFORE doTwo() and like: {5, 200, 15, 20, 25} IF CALLED AFTER

The program contains the following methods:

```
static void doOne(int n){
   n=n*10;
}


static void doTwo(int[] arr, int j){
   arr[j] = arr[j+2] *10;
}


static void doThree(int[] arr){
   for (int i=0; i<arr.length; i++)
      arr[i] = arr[i]*5;
}
```

Question 6. (12 points – 4 points each)

a. An array, **numbers**, of integers is filled with 100 random numbers whose values range from 10 to 199 (inclusive). You DO NOT have to write code to generate the random numbers. Write the code to count how many elements of the array are even and between 30 and 40 (inclusive).

```
int count = 0;
for (int i = 0; i < numbers.length; i++){
        if(numbers[i] >= 30 && numbers[i] <= 40){
                count++;
        }
}
```

b. The quadratic formula can be used to solve for the two values of x in a quadratic equation of the form $ax^2+bx+c=0$. Write an expression in Java that will compute one of the values of x based on the formula below:

$$\frac{-b+\sqrt{b^2-4ac}}{2a}$$

```
        double root1, root2;

        // calculate the determinant (b2 - 4ac)
        double determinant = b * b - 4 * a * c;

        // check if determinant is greater than 0
        if (determinant > 0) {

                // two real and distinct roots
                root1 = (-b + Math.sqrt(determinant)) / (2 * a);
                root2 = (-b - Math.sqrt(determinant)) / (2 * a);

                System.out.printf("root1 = %.2f and root2 = %.2f", root1,
root2);
        }

        // check if determinant is equal to 0
        else if (determinant == 0) {

                // two real and equal roots
                // determinant is equal to 0
                // so -b + 0 == -b
                root1 = root2 = -b / (2 * a);
                System.out.printf("root1 = root2 = %.2f;", root1);
        }

        // if determinant is less than zero
        else {

                // roots are complex number and distinct
                double real = -b / (2 * a);
                double imaginary = Math.sqrt(-determinant) / (2 * a);
                System.out.printf("root1 = %.2f+%.2fi", real, imaginary);
                System.out.printf("\nroot2 = %.2f-%.2fi", real, imaginary);

        }
```

c. Suppose I am doing a binary search on the following array for the number 1

int nums[ ] = {5, 77, 78, 89, 100, 117, 125, 235, 390, 1000};

List the high, low and midpoint values at each step until 1 is not found. DO NOT write any code. Enter into the table below the low, mid and high values as the binary search progresses. You may use either the subscript (index) values or the actual number stored at the subscript. Not all rows may be needed.

Low	**Mid**	**High**
5 (index 0)	100 (index 4)	1000 (index 9)
5 (index 0)	77 (index 1)	89 (index 3)
5 (index 0)	5 (index 0)	5 (index 0)

```
binarySearch(arr, x, low, high)
    repeat till low = high
    mid = (low + high)/2
    if (x == arr[mid])
        return mid

    else if (x > arr[mid]) // x is on the right side
        low = mid + 1

    else                // x is on the left side
        high = mid - 1
```

Question 7. (30 pts)

Write a complete Java program with comments in main and in each method.

Data: The input data for this program is given as two columns of numbers. All data will be entered from a file named **input.txt** and all output will go to the screen. Assume there will not be more than 100 rows of data in the file.

Sample Data Set:

```
7 23.56
16 88.12
10 75.1
```

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

1) Reads the data into two arrays of doubles, arr1 and arr2, by invoking the method readData. with both arrays as parameters. The return value from readData should be stored in an integer variable num.
2) Modifies all of the values in the arr2 array by invoking the method changeArray, passing both arrays and num as parameters.
3) Sort arr1 and arr2 separately by invoking the method sortArray twice – once for arr1 and once for arr2
4) Prints the values in both arrays to the screen in two columns as shown below with a header and the numbers right aligned

```
 arr1    arr2
 7.00  164.92
10.00  751.00
16.00 1409.92
```

Method Details:

The readData method reads two doubles values from the input file. The first value of each line is stored in arrOne, and the second value is stored in arrTwo. For example, in the Sample Data above, 7 would go into arrOne[0] and 23.56 would go into arrTwo[0]. This is repeated until there are no more values in the file.
Parameters:
  arrOne - an array of doubles for storing the 1st value read in per row
  arrTwo – an array of doubles for storing the 2nd value read in per row
Returns:
  num - an integer representing the number of rows that were read in

The changeArray method has parameters:
  alpha – an array of doubles
  beta – an array of doubles
  k - an integer representing the number of elements in alpha and beta to be processed

The method multiplies each element of beta by the corresponding value of alpha and stores the result into beta. For example, beta[0],7, is multiplied by alpha[0], 23.56 and the result, 164.92, is stored in beta[0]
.

The sortArray method sorts the first k elements of the parameter array into ascending order.
Parameters:
  arrToOrder - an array of doubles to be sorted
  k - an integer representing the number of elements in the array arrToOrder