CISC 1115 Fall 2017 Sample Exam

Question 1.

Each of the following sections of code doesn't work properly. In each case, show how to fix the code so that it works properly. There is one error in each section of code. Put your answer in the box beneath the question.

a. The following section of code is supposed to calculate the sum of the numbers 1 to n. Assume that sum and n have been declared and n has been assigned a value.

```
for (int i = 1; i <= n; i++){
        sum = 0; //does not work because it resets the value of sum for each iteration of the loop, so instead of getting the sum of 1 - 10, we always just end up adding the current value of i to 0, should be moved outside of (above) the loop
        sum = sum + i; //adds to sum, ideally should work as: 0 + 1 + 2 + ... + 10
}
System.out.println("sum = " + sum); //prints out sum, ideally should print: sum = 55
```

should be:

```
sum = 0;
for (int i = 1; i <= n; i++){
        sum = sum + i; //adds to sum, ideally should work as: 0 + 1 + 2 + ... + 10
}
System.out.println("sum = " + sum); //prints out sum, ideally should print: sum = 55
```

b. The following section of code is supposed to print each of the numbers from 1 to n, each on a new line. (Assume that n has been assigned a value.)

```
for (int i = 1; i <= n; i++)
        System.out.println(n); //prints out value on new line each, does not work because n is the set value (not incremented); if we want to print out the values from 1 to n we should print out i
```

should be:

```
for (int i = 1; i <= n; i++)
        System.out.println(i);
```

c. The following section of code is supposed to call the method doIt repeatedly as long as the user wants to continue.

```
char answer;
Scanner kybd = new Scanner(System.in);
do {
        doIt(); //calls doIt method
        System.out.print("Do you want to continue? "); //prompts the user to ask if to continue
        answer = kybd.next().charAt(0);
} while (answer == 'y' && answer == 'Y'); //does not work because it uses and instead of or, when using and in a condition, all parts of the condition have to be true for the condition to evaluate as true. There is no case where this condition can evaluate to true as answer can only be one character at a time and 'y' and 'Y' are separate characters that are not equivalent in Java
```

should be:

```
char answer;
Scanner kybd = new Scanner(System.in);
do {
        doIt();
        System.out.print("Do you want to continue? ");
        answer = kybd.next().charAt(0);
} while (answer == 'y' || answer == 'Y');
```

d. The following method is supposed to check whether any of the values in the array are negative.

```
public static boolean hasNegValue(int[] a, int n) {
        for (int i = 0; i < n; i++)
                if (a[i] < 0)
                        return true;
                else //does not work, puts the return false in the scope of the for loop
                        return false;//should only return false AFTER for loop has found no
                negative numbers
 }//method missing return statement since the returns are INSIDE the scope of the for loop
```

should be:

```
public static boolean hasNegValue(int[] a, int n) {
        for (int i = 0; i < n; i++)
                if (a[i] < 0)
                        return true;

        return false;
}
```

Question 2.

Part A.

Given the following method:

```
int methodA(int x, int y) {
    if (x < y)
        return -1;
    else if (x== y)
        return 0;
    else
        return 1;
}
```

What would be returned for the following calls:

i. methodA(3, 3) ___**0**___

ii. methodA (5, 4) ___**1**___

iii. methodA (20, 30) ___**-1**___

iv. In general, what does the method return?

This method returns:

- **-1 if the value of X is less than the value of Y**
- **0 if the values of X and Y are equal**
- **1 if the value of X is greater than the value of Y**

v. Are there any other cases to take into consideration that are not covered by the above three calls? Answer either “No”, or provide such a case.

No

vi. Rewrite the method eliminating the if statements; instead use conditional operators only

return x < y ? -1 : x == y ? 0 : 1;

Part B.

Given the following method:

```
String restOf(String s) {
        return s.substring(1);
}
```

What is returned by each of the following expressions:

i. "J" + restOf("Hello") _____**Jello**_____

"ello"

ii. restOf(restOf(restOf("cowboy"))) ___**boy**____

restOf(restOf("owboy")))

restOf("wboy")

"boy"

Question 3.

Part A:

A String variable sentence consists of words separated by single spaces.

Write the Java code needed to print the number of words that start with an upper case letter. So for "The deed is done", your code would print 1 but for "The name is Bond, JAMES Bond", your code would print 4.

```
int count = 0;
if(Character.isUpperCase(sentence.charAt(0))){//if the first letter is upper case
        count++;//count it!
}

int loc = sentence.indexOf(" ");//finds the index of the first space
while (loc != -1) {//as long as there are spaces in the sentence
        if(Character.isUpperCase(sentence.charAt(loc + 1))){//if the first letter of the next word (after the space) is upper case
                count++;//count it too!
        }
                loc = sentence.indexOf(" ", loc + 1);//continue updating the value for loc based on the next location of a space
}

System.out.println("There are " + count + " words that start with an uppercase letter.");
```

Part B:

An int variable n contains a positive integer and a String variable s contains a String. Write the Java code needed to print "YES" if the first n characters of s appear more than once in s, and print "NO" otherwise.

```
String s2 = s.substring(0, n);//finds the first n (#) characters in String s

if(s.substring(1).contains(s2)){//if that sequence of characters appear AGAIN in the String
   System.out.println("Yes");
}
else {
   System.out.println("No");
}
```

Part C:

Show how to invoke the method grunf so that it will return the String HOPE.

```
public static String grunf(String s1, String s2, String s3) {
        String s = s1.substring(1,2);
        if (s1.length()>s2.length() && s2.compareTo(s3)<0
                        && s3.indexOf(s)>0)
                return s1.substring(0,1) + s2.substring(1);
        else
                return s1.substring(0,1) + s2.substring(0,1) +
                                s1.substring(0,1);
}
```

String result = grunf("HURRY", "COPE", "QUERY");
String result = grunf("HOBBLE", "HOPE", "SO");

Question 4.

a. Suppose an exam grade called **grade** (integer) is considered **invalid** if it is either negative or above 100. Write code in Java that will set the variable **invalid** to true if the value of **grade** is invalid and false otherwise.

```
boolean invalid;

if(grade < 0 || grade > 100) //grade (integer) is considered invalid if it is either negative or above 100
        invalid = true; //set the variable invalid to true if the value of grade is invalid
else
        invalid = false; //false otherwise
```

b. Write a section of code in Java that will do the following: Generate 10 random numbers (integers) each between 1 and 10. Print the number of random numbers that are greater than 5.

```
int count = 0;

for(int i = 1; i <= 10; i++){
        Random random = new Random();
        int n = random.nextInt(10) + 1; //nextInt(10) generates numbers from [0 … 9] so +1 generates [1 … 10]
        if(n > 5)
                count++;
}

System.out.println("There are " + count + " numbers greater than 5");
```

c. Consider an array of 10 integers that contains the following values in increasing sequence:

3 15 25 43 50 76 88 102 110 120

Assume you wish to find the location of the element with value 43.

i. Show the sequence of numbers that you would encounter in this search if you used a **sequential search**.

3 → 15 → 25 → 43 //FOUND

ii. Show the sequence of numbers that you would encounter in this search if you used a **binary search**.

L → 3
M → 50 //43 is less than 50 so we will continue to search in the LEFT side of the array
H → 120

L → 3
M → 15 //43 is greater than 15 so we will continue to search in the RIGHT side of the array
H → 43 //mid - 1

L → 25 //mid + 1
M → 25 //43 is greater than 25 so we will continue to search in the RIGHT side of the array
H → 43

L → 43 //mid + 1
M → 43 //FOUND
H → 43

d. Suppose you wish to compute the value of $a^2 + b^3$ for some integers a and b. Write a statement in Java that would place the value of this expression in an integer variable **x**.

```
int x = Math.pow(a, 2) + Math.pow(b, 3); //a^2 + b^3
```

Question 5.

Write a complete Java program to do the following:

a. Read from a file **input5.txt** a list of pairs of integers. For each pair, if the sum of the two numbers is positive, calculate as a decimal number the first divided by the second. If the sum is negative, calculate as a decimal number the second divided by the first. You may assume neither value is zero.

b. Write the two numbers and the calculated quotient to a file **output5.txt,** one pair per line. Do this until there is no more data to read. (EOF)

c. At the end, print 2 things with an appropriate message: The number of pairs processed, and the value of the quotient closest to zero. Remember that the quotient may be positive or negative.

```
import java.util.*;

import java.io.*;

public class Question5 {
	public static void main (String [] args) throws Exception {
		PrintWriter output = new PrintWriter (“output5.txt”):
		File input = new File (“input5.txt”);
		Scanner sc = new Scanner (input);
		int numPairs = 0;
		double nearestToZero = 0;
		while(sc.hasNext()) { //read until EOF
			int n1 = sc.nextInt();
			int n2 = sc.nextInt();
			double quotient;
			if (n1 + n2 >= 0) { //if the sum of the two numbers is positive
				quotient = (double) n1 / n2;
			}
			else { //implied that if the sum is not positive, it is negative
				quotient = (double) n2 / n1;
			}

			output.println(n1 + “ “ + n2 + “ Quotient: “ + quotient);
			numPairs++;//count each pair processed

			// by default first is nearest or it will check for other numbers
				if (numPairs == 1 || Math.abs(quotient) <
Math.abs(nearestToZero))
				nearestToZero = quotient;

		}

		output.println(“Number of pairs processed: “ + numPairs);
		output.println(“Quotient nearest to zero: “ + nearestToZero);

		output.close();
	}
}
```

Question 6.

Perform the following conversions:

a. 111001 (base 2) to base 10 Answer: _____**57**_____

111001 >> 6 digits >> base 2^5 - 2^0

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

32 + 16 + 8 + 0 + 0 + 1 = 57

b. C4 (base 16) to base 10 Answer: ___**196**___

C4 >> 2 digits >> base 16^1 - 16^0

C = 12

(12 * 16^1) + (4 * 16^0)

192 + 4 = 196

c. 1101001 (base 2) to base 16 Answer: ____**69**____

1101001

110 1001

0110 = 6

1001 = 9

d. 34(base 10) to base 2 Answer: ___**100010**____

34 / 2 = 17 **remainder = 0** **(also can take modulus, i.e. 34 % 2)**
17 / 2 = 8 **remainder = 1**
8 / 2 = 4 **remainder = 0**
4 / 2 = 2 **remainder = 0**
2 / 2 = 1 **remainder = 0**
1 / 2 = 0 **remainder = 1**

1 0 0 0 1 0

e. E8 (base 16) to base 2 Answer: ___**11101000**____

E = 1110

8 = 1000

1110 1000

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

101 + 111 = ? Answer: ____**1100**____

//5 + 7 = 12

Question 7.

For this question, assume all input comes from the keyboard, and all output goes to the screen. Include method prototypes and comments. The array should have room for 100 integers.

Write a complete Java program, including at least one comment in the main program and one in each method, to do the following:

Write a main program which will call the methods described below.

(a) First the main program will read an integer (this integer is a parameter or header value) which it will call **k**. Then the main program will call the method **readdata()** (see part 1 below) to read a set of data containing **k** values into an integer array which the main program will call **x**.

(b) The main program will call the method **count5s()** (see part 2 below), sending the **x** array (and **k**).

(c) The main program will call the method **sortarray()** (see part 3 below) to sort the first **k** elements of the **x** array into descending order. The new values in the **x** array will be printed in the main program.

```
import java.util.*;

import java.io.*;

public class Question7 {

    public static void main (String [] args) {

        Scanner sc = new Scanner (System.in);

        //part a

        int k = sc.nextInt(); //read an integer (this integer is a parameter/header value) which it will call k

        int [] x = new int [100]; //integer array which the main program will call x, should have room for 100 integers

        sc.close();

        readdata(k, x);

        //part b

        count5s(k, x);

        //part c

        sortarray(x, k);

        for(int i = 0; i < k; i++)

            System.out.print(x[i] + " "); //prints the sorted array

    }
```

Details of the methods:

1. Write a method called **readdata()** which receives two parameters, an integer **n** and an integer array **arr**.

The method will be sent a value for **n**. The method will read a set of **n** data items into the **arr** array. The method will print the set of data values as they are being read in.

```
public static void readdata(int n, int [] arr){
    Scanner sc = new Scanner (System.in);
    for(int i = 0; i < n; i ++){ //n data items to read in
        arr[i] = sc.nextInt();
        System.out.print(arr[i] + " "); //print the set of data as they are being read in
    }
    sc.close();
}
```

2. Write a method called **count5s()** which receives two parameters, an integer **n** and an integer array **v**. The method will count how many of the first **n** elements of the **v** array are greater than 5, how many are less than 5, and how many are equal to 5.

The method will print these values, with messages for each.

As an example: if the array holds the values 5 66 -4 5 1 -3, with n = 6: there is 1 value greater than 5, there are 3 values less than 5, and there are 2 values equal to 5.

```
public static void count5s(int n, int [] v){
    int greaterThanFive = 0;
    int lessThanFive = 0;
    int equalToFive = 0;
    for(int i = 0; i < n; i++){
        if(v[i] > 5) //if element is greater than 5
            greaterThanFive++; //count
        else if (v[i] < 5) //if element is less than 5
            lessThanFive++; //count
        else //implied that if not greater or less than, then element is equal to 5
            equalToFive++; //count
    }
    System.out.println("There are " + greaterThanFive + "values greater than 5.");
    System.out.println("There are " + lessThanFive + "values less than 5.");
    System.out.println("There are " + equalToFive + "values equal to 5.");
}
```

3. Write a method called **sortarray()** which receives two parameters, an integer array **w** and an integer **n** giving the size of the array. The method will sort the first **n** elements of the **w** array into descending order.

As an example, assume that method starts with these values in the **w** array: 5 66 -4 5 1 -3, with n = 6. Inside the method, these values will be sorted into descending order; the array will now hold 66 5 5 1 -3 -4.

```
public static void sortarray(int [] w, int n){ //bubble sort
    for (int i = 0; i < n-1; i++){
        for (int j = 0; j < n-i-1; j++){
        if (w[j] < w[j+1]){ //descending order
            int temp = w[j];
            w[j] = w[j+1];
            w[j+1] = temp;
        }
        }
    }
}
```