QUESTION 1

a) Write Boolean expressions for the following (where H, I, J, K, L, M, N are integers):

i. A is a negative odd number.

(A%2==1 && A<0) (2 marks)

ii. E is a multiple of F.

(E%F==0) (2 marks)

iii. The sum of E and F is greater than the product of G and H.

((E + F) > (G * H))   (2 marks)

iv. D is less than 30, E is not less than 0 and F is equal to 64.

(D<30 && E>=0 && F==64) (2 marks)

v. M is not the square of P

(M! = P*P) (2 marks)

(10 marks)

b) Please explain the difference between the following 2 statements.

(10 marks)

```
int x = 4;

x = 8;
```

```
int y = 15;

boolean isTen = (y == 15);
```

The first box creates an int called x and initialises it to 4, then changes its value to 8 **(4 marks)**

The second box creates an int called y and initialises it to 15, then checks to see if the value of y is equal to 15 **(6 marks)**

c) The library method Math.random() yields a real number in the range
$0.0 \leq$ Math.random() < 1.0
Using this method, write down a Java expression denoting a random integer in the range 37..84 inclusive.

(5 marks)

```
(int) ( Math.random( ) *48) + 37;
```

0 $\leq$ (int) (Math.random( ) * 48) $\leq$ 47

37 $\leq$ (int) (Math.random( ) * 48) + 37 $\leq$ 84

2 marks for the correct minimum number

2 marks for the correct maximum number

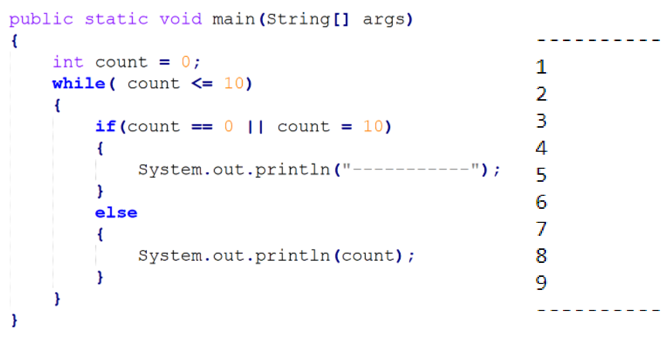
1 uses Math.Random

d) Determine the value of the variables in the statement **product *= x++;** after the calculation is performed. Assume that product and x are type int and initially have the value 5.

(5 marks)

Answer: product = 25 (3 marks), x = 6 (2 marks)

e) Explain exactly why this code will not work as intended. What two fixes are needed to get the output on the right?



```
public static void main(String[] args)
{
    int count = 0;
    while( count <= 10)
    {
        if(count == 0 || count = 10)
        {
            System.out.println("----------");
        }
        else
        {
            System.out.println(count);
        }
    }
}
```

```
----------
1
2
3
4
5
6
7
8
9
----------
```

(10 marks)

Answer:

Marks (5)

Error one: count is not incrementing **(2)**
Correction: add count++ at the end of the while loop **(2)**
Error two: count = 10 is doing an assignment not a comparison **(4)**
Correction: the comparison needs to be if(count ==0 || count == 10) **(2)**

Total (40 marks)

SECTION B – <u>FOUR QUESTIONS TO BE ATTEMPTED</u>

QUESTION 2

Write a method called **Password** which checks if the passed strings is a strong password The rules for a strong password are:

- Its length is at least 12.
- It contains at least one digit.
- It contains at least one lowercase English character.
- It contains at least one uppercase English character.
- It contains at least one special character. The special characters are: !@#$%^&*()-+

The method should return a Boolean value based on the check performed.

Total (15 marks)

Solution:

```java
public static boolean strongPassword (String a) {
        boolean length = false;
        boolean lowercase = false;
        boolean uppercase = false;
        boolean special = false;
        String sp = "!@#$%^&*()-+";

        if (a.length()>=12){
            length = true;
            for (int i = 0; i < a.length(); i++) {
                char c = a.charAt(i);
                if (c >='a' && c<='z')  lowercase = true;
                if (c >='A' && c<='Z')  uppercase = true;
                if (sp.indexOf(c)>=0)   special = true;
            }
        }

        return (length && lowercase && uppercase && special);

    }
```

Total (15 marks)

Marks (15)

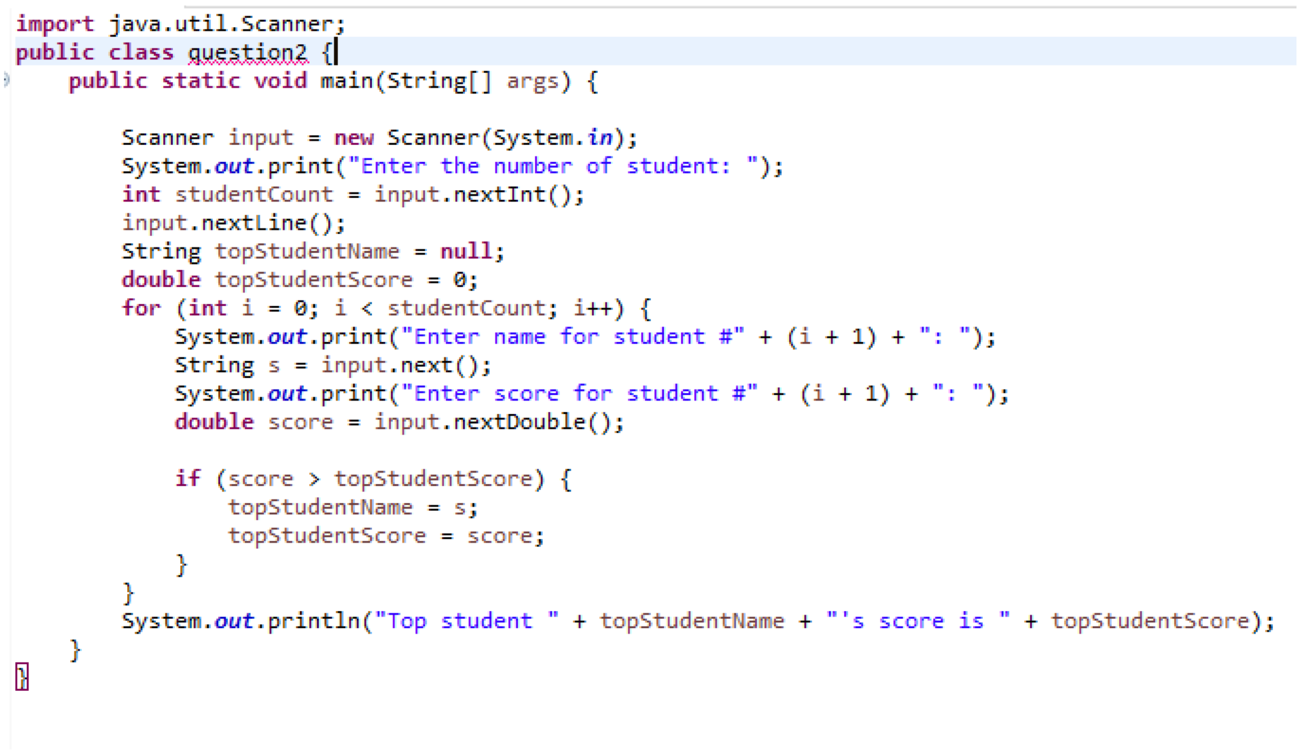
Defining Variables (**2 marks)**
There should be four "conditional" statements, each worth two marks, for a total of 8 marks. These statements may be separated. **(8 marks)**
For loop **(3 marks)**
Return statement **(2 mark)**

QUESTION 3

Write a program that prompts the user to enter the number of students, each student's name and score, and finally displays the name of the student with the highest score.

(15 marks)

Solution:



```java
import java.util.Scanner;
public class question2 {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        System.out.print("Enter the number of student: ");
        int studentCount = input.nextInt();
        input.nextLine();
        String topStudentName = null;
        double topStudentScore = 0;
        for (int i = 0; i < studentCount; i++) {
            System.out.print("Enter name for student #" + (i + 1) + ": ");
            String s = input.next();
            System.out.print("Enter score for student #" + (i + 1) + ": ");
            double score = input.nextDouble();

            if (score > topStudentScore) {
                topStudentName = s;
                topStudentScore = score;
            }
        }
        System.out.println("Top student " + topStudentName + "'s score is " + topStudentScore);
    }
}
```

Marks (15)

Getting number of students with nextInt **(3 marks)**
For loop until studentcount **(3 marks)**
Input.next() to get student name **(2 marks)**
Input.nextDouble() to get student score **(2 marks)**
If to check the score **(4 marks)**
Printing top student result **(1 mark)**

QUESTION 4

(a) Create a method named *insertionSort*. The method should take an integer array as an input parameter. The method should end up creating and returning a new array that contains the values of the original array sorted in ascending order. This method should use the Insertion Sort algorithm to sort these values. The original array, which was passed as an input parameter, should be left unchanged.

(10 marks)

Sample Solution:

```
public static int [] insertionSort(int[] origArray){
      // Copy the original array to a new array a.
      int [] a = new int [origArray.length];
      for (int i=0; i<origArray.length; i++)
            a[i] = origArray[i];

      // sort the new array using insertion sort
      for(int j=1;j<a.length;j++){
            int t = a[j];
            int i = j-1;
            while(i>=0&&a[i]<t){
                  a[i+1] = a[i];
                  i--;
            }
            a[i+1] = t;
      }
      return a;
}
```

10 marks:

- Correct method signature with int array argument and return type **(1.5)**
- Copying original array and working with and returning the copy **(2)**
- Outer loop **(1.5)**
- Picking next item from the unsorted area to insert **(1)**
- Inner loop correctly shifts items in sorted area **(2)**
- Inner loop inserts new item into correct position in sorted area **(2)**

(b) Create a method called *hasDuplicateElements*. The method should take an integer array as an input parameter. The method should test whether the integer array contains duplicate integers, returning true if it does and returning false if all of its integers are unique. Your method should invoke and use the "insertionSort" method from part a.

(5 marks)

5 marks:

- Correct method signature **(1 mark)**
- Appropriate use of the insertionSort method **(1 mark)**
- Loop to check for adjacent duplication or other suitable approach **(3 marks)**

Note: *Any correct solution that does not use insertionSort but rather utilizes some alterative approach should still get 4 out of 5 marks.*

Sample Solution:

```
public static boolean hasDuplicateElements(int[] a){
      // get a sorted version of the array
      int [] sortedArray = insertionSort(a);

      // any non-unique elements will be adjacent
      for (int i=0; i<sortedArray.length-1; i++)
            if (sortedArray[i]==sortedArray[i+1])
                  return true;
      return false;
}
```

QUESTION 5

Write a Java program to read a text file called list.txt and write another text file called distribution.txt. Each line of the input file consists of a list of numbers from 0 to 9, for example:

```
01223334444566

02134585355201
```

The lines have no extra spaces or tabs at the beginning or end. The program should write only the distribution of each number in the input file using asterisks (stars) to represent the count of each number, as shown in the example below. Ensure to include exception handling in your program for cases where the file is not found or other input/output errors occur.

```
0:***

1:***

2:****

3:*****

4:*****

5:*****

6:**

7:

8:*

9:
```

Total (15 marks)

```
import java.util.*;

import java.io.*;


class Q52{

        public static void main (String[] args){

                        // Create Scanner object

                        int[] dist = new int[10];

                        try{

                                // Create Scanner object for file

                                Scanner inFile = new Scanner(new File("list.txt"));

                                PrintWriter outFile = new PrintWriter("distribution.txt");


                                while (inFile.hasNextLine())   // any numbers left?

                                {

                                 String s = inFile.nextLine();     // read number

                                 for(int x=0; x<s.length();x++){

                                        char c = s.charAt(x);

                                        int t = c-'0';

                                        dist[t]++;


                                 }

                                }

                                inFile.close();

                                for (int i = 0; i<dist.length; i++){

                                        String a = " : ";

                                        for(int j =0;j<dist[i];j++){

                                                a = a+"*";

                                        }

                                        outFile.println(i + a);
```

```
                }

            outFile.close();  // Important!

            }
            catch (FileNotFoundException e)
            {
                System.out.println(e.toString());
            }
            catch (Exception e){
                System.out.println ("Uh Oh Something went wrong");
                System.out.println(e.toString());
            }
        }
}
```

15 marks:

- Creating Scanner and PrintWriter objects (2 marks)
- Reading from the file and processing input (4 marks)
- Writing to the output file (4 marks)
- Exception handling (3 marks)
- Code structure and closing resources (2 marks)

QUESTION 6

Create a class called *Invoice* that a hardware store might use to represent an invoice for an item sold at a store. An Invoice should include four pieces of information as instance variables—a part number (type String), a part description (type String), a quantity of the item being purchased (type int), and a price per item (double). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named *getInvoiceAmount* that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0.

(15 marks)

Solution:

```
public class Invoice {
    private String number;
    private String description;
    private int quantity;
    private double pricePerItem;

    public Invoice(String number, String description, int quantity, double pricePerItem) {
        this.number = number;
        this.description = description;

        if (quantity > 0) { (2 mark)
            this.quantity = quantity;
        }

        if (pricePerItem > 0.0) { (2 mark)
            this.pricePerItem = pricePerItem;
        }
    }

    public double getInvoiceAmount() { (1 mark)
        return quantity * pricePerItem;
    }

    public String getNumber() { (1 mark)
        return number;
    }

    public void setNumber(String number) { (1 mark)
```

```
		this.number = number;
	}

	public String getDescription() { **(1 mark)**
		return description;
	}

	public void setDescription(String description) { **(1 mark)**
		this.description = description;
	}

	public int getQuantityPurchased() {**(1 mark)**
		return quantity;
	}

	public void setQuantity(int quantity) {**(2 mark)**
		if (quantity > 0) {
			this.quantity = quantity;
		}
	}

	public double getPricePerItem() { **(1 mark)**
		return pricePerItem;
	}

	public void setPricePerItem(double pricePerItem) { **(2 marks)**
		if (pricePerItem > 0.0) {
			this.pricePerItem = pricePerItem;
		}
	}
```

}

Marks: 15 (marks breakdown in the solution program)

(2 mark) if (quantity > 0)

(2 mark) if (pricePerItem > 0.0)

(1 mark) public double getInvoiceAmount()

(1 mark) public String getNumber()

(1 mark) public void setNumber(String number)

(1 mark) public String getDescription()

(1 mark) public void setDescription(String description)

(1 mark) public int getQuantityPurchased()

(2 mark) public void setQuantity(int quantity)

(1 mark) public double getPricePerItem()

(2 marks) public void setPricePerItem(double pricePerItem)