START WRITING FROM HERE

3. Exercises on decision and loop

3.1 SumAverageRunningInt (Decision & loop)

Write a program called SumAverageRunningInt to produce the sum of 1, 2, 3 ..., to 100. Store 1 and 100 in variables lowerbound and upperbound, so that we can change their values easily. Also compute and display the average.

```
public class SumAverageRunningInt {
    public static void main (String args[]) {
        int lowerbound=1, upperbound =100, sum = 0;
        for (int i = lowerbound; i<= upperbound ; i++){
            sum += i;
        }
        float average = sum / (upperbound - lowerbound +1);
        System.out.println ("The sum of " + lowerbound +
                            "to " + upperbound + "is" + sum);
        System.out.println ("The average is " + average);
    }
}
```

output of the above code :

```
The sum of 1 to 100 is 5050
The average is 50.5
```

ROLL NUMBER :

1. Modify the program to use a "while" loop instead of a "for" loop.

```
public class SumAverageRunningInt {
    public static void main(String args[]) {
        int lowerbound = 1, upperbound = 100, sum = 0;
        int i = lowerbound;
        while (i <= upperbound) {
            sum += i;
            i++;
        }
        double average = (double) sum / (upperbound - lowerbound + 1);
        System.out.println("The sum of " + lowerbound + " to " +
                           upperbound + " is " + sum);
        System.out.println("The average is " + average);
    }
}
```

2. Modify the program using do-while loop.

```
public class SumAverageRunningInt {
    public static void main(String args[]) {
        int lowerbound = 1, upperbound = 100, sum = 0;
        int i = lowerbound;
        do {
            sum += i;
            i++;
        } while (i <= upperbound);
```

ROLL NUMBER :

```
        double average = (double) sum/(upperbound - lowerbound + 1);
        System.out.println("The sum of " + lowerbound + "to" +
                                upper bound + "is" + sum);
        System.out.println("The average is " + average);
    }
}
```

3. What is the difference between "for" and "while-do" loops ? What is the difference between "while-do" loops and "do-while" loops?

	for loop	while loop
1.	Initialization may be either in the loop statement or outside the loop.	Initialization is always outside the loop.
2.	Increment is done in loop statement.	Increment is done inside the loop.
3.	used when number of iteration is known.	used to when number of iterations is unkown.

4.	Syntax:
	for(init ; condition ; iteration){
	statements ;
	}
		Syntax :
		while (condition) {
		statements ;
		}

Difference between while and do-while loops

	while loop	do-while loop
1.	while loop is entry control loop.	Do-while is exit control loop.

2.	If condition is initially false, loop never runs	Even if the condition is false, do-while loop is executed at least once.

3.	Syntax:
	while (condition) {
	statements;
	}
		Syntax:
		do {
		statements;
		} while (condition);

4. Modify the program to sum from 111 to 8899, and compute the average. Introduce an int variable called count to count the numbers in the specified range (to be used in computing the average).

```
public class SumRunningAverageInt {
    public static void main (String args[]) {
        int lowerBound = 111, upperBound = 8899, sum = 0, count=0;
        for (int i = lowerBound; i <= upperBound; i++) {
            sum += i;
            count++;
        }
        double average = (double) sum / count;
        System.out.println ("The sum of" + lowerBound + "to" +
                            upperBound + "is" + sum);
        System.out.println ("The average is" + average);
    }
}
```

Output :

The sum of 111 to 8899 is 395 9445

The average is 4505.0

ROLL NUMBER :

5. Modify the program to find the "sum of the squares" of all the numbers from 1 to 100. i.e 1*1 + 2*2 ....

```
Public class SumAverageRunningInt {
    public static void main (String args[]){
        int lowerBound = 1, upperBound = 100, sumOfSquares = 0;
        for (int i = lowerBound; i <= upperBound ; i++ ){
            sumOfSquares += (i*i);
        }
        System.out.println ("The sum of the squares is " + sumOfSquares);
    }
}
```

output :

The sum of the squares is 338350.

6. Modify the program to produce two sums : sum of odd numbers and sum of even numbers from 1 to 100. Also compare their absolute difference.

```
public class SumAverageRunningInt {
    public static void main (String args[]){
        int lowerBound = 1, upperBound = 100, sumOfEven = 0,
            sumOfOdd = 0;
        for (int i = lowerBound; i <= upperBound ; i++ ) {
            if (i%2 == 0) {
                sumOfEven += i;
            }
            else {
                sumOfOdd += i;
            }
        }
```

```
        System.out.println ("sum of even squares : " + sumOfEven);
        System.out.println ("sum of odd numbers : " + sumOfOdd);
        System.out.println ("absolute difference : " + (sumOfEven - sumOfOdd));
    }
}
```

3.2 production (or factorial) (Decision & Loop)

Write a program called Product1ToN to compute the product of integers from 1 to 10 (i.e. 1x2x3x...x10) as an int. Take note that it is the same as factorial of N.

```
public class Product1ToN {
    public static void main (String args[]){
        int product = 1;
        int LowerBound = 1;
        int UpperBound = 10;

        for (int i = LowerBound; i <= UpperBound; i++){
            number *= i;
        }
        System.out.println ("factorial is : " + number);
    }
}
```

Output :

factorial is : 3628800

1. Compute the product from 1 to 11, 1 to 12, 1 to 13 and 1 to 14. Write down the producer obtained and decide if the results are correct

ROLL NUMBER :

2. Repeat the above, but use long to store the product. Compare the products obtained with int for N = 13 & N = 14.

N	with int	with long
11	39916800	39916800
12	479001600	479001600
13	1932053504	6227020800
14	1278945280	87178291200

When using int 13! and 14! go out of range of integer variable but using long, the value is in range. Thus the difference in output.

3.3 HarmonicSum (Decision & Loop)

Write a program called HarmonicSum to compute the sum of a harmonic series, as shown below, where n = 50000. The program shall compute the sum from left to right as well as from the right to left. Are the two sums the same? Obtain the absolute difference between these two sums and explain the difference. Which sum is more accurate?

$$Harmonic(n) = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$$

```
public class HarmonicSum {
    public static void main (String args[]){
        final int maxDenominator = 50000;
        double sumL2R = 0.0;
        double sumR2L = 0.0;
        double absDiff;
        for (int i =1; i <= maxDenominator; i++){
            sumL2R += (double) (1) /i;
        }
        for (int j = maxDenominator; j>=1; j--){
            sumR2L += (double) (1) / j ;
        }
        System.out.println ("the sum from left to right: " + sumL2R);
        System.out.println ("the sum from right to left: " + sumR2L);
        if (sumL2R > sumR2L){
            absDiff = sumL2R - sumR2L;
        }
        else{
            absDiff = sumR2L - sumL2R;
        }
        System.out.println ("absolute difference: " + absDiff);
    }
}
```

output :

the sum from left to right : 11.397003949278504

the sum from right to left : 11.397003949278519

absolute difference : 1.4210854715202004 E -14

3.4 ComputePI (Decision & Loop)

Write a program called ComputePI to compute the value of π, using the following series expansion. Use the max. denominator as the terminating condition. Try maxDenominator of 1000, 10000, 100 000, 10 00 000 and compare the Pi obtained. Is this series suitable for computing Pi? Why?

$$\pi = 4 \times \left[1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} - \frac{1}{15} + \cdots \right]$$

```
public class ComputePI {
    public static void main (String args[]){
        double sum = 0.0;
        int maxDenominator = 1000;
        for ( int denominator =1; denominator <= maxDenominator;
            denominator +=2 ){
            if (denominator % 4 == 1 ) {
                sum += (double) (1) / denominator;
            } else if (denominator % 4 == 3 ) {
                sum -= (double) (1) / denominator;
            } else {
                System.out.println ("Impossible!!!");
            }
        }
        double Pi = (double) (4)* (sum);
        System. out. println ("Pi value:" + pi );
    }
}
```

ROLL NUMBER :

output :

maxDenom	Pi value
1000	3.139592655589795
10000	3.141392653591791
100000	3.141572653589793
10 00 000	3.141590653589692

1. Instead of using maximum denominator as the terminating condition, rewrite your program to use the max no. of terms (maxTerm) as the terminating condition.

```
public class ComputePI {
    public static void main (String args[]) {
        int maxTerm = 10000;
        double sum = 0.0;
        for (int term = 1 ; term <= maxTerm ; term++ ) {
            if (term % 2 == 1) {
                sum += (double) (1) / (term*2 - 1);
            }
            else {
                sum -= (double) (1) / (term*2 - 1);
            }
        }
        double Pi = (double) (4) * sum;
        System.out.println ("Pi value: " + Pi);
        System.out.println ("comparison: " + (Pi / Math.PI) * 100));
    }
}
```

ROLL NUMBER :

3.5 CozaLozaWoza (Decision & Loop)

write a program called CozaLozaWoza which prints the number 1 to 110, 11 numbers per line. The program shall print "Coza" in place of the numbers which are multiples of 3, "Loza" for multiples of 5, "Woza" for multiples of 7, "CozaLoza" for multiples of 3 and 5, and so on. The output shall look like:

1 2 Coza 4 Loza Coza Woza 8 Coza Loza 11
Coza 13 Woza CozaLoza 16 17 Coza 19 Loza CozaWoza 22
23 Coza Loza 26 Coza Woza 29 CozaLoza 31 32 Coza
.....

```
public class CozaLozaWoza {
    public static void main (String args[]){
        final int lowerBound = 1; upperBound = 110;
        for (int i = lowerBound; i <= upperBound; i++){
            if (i%3 == 0){
                System.out.print("Coza");
            }
            if (i%5 == 0){
                System.out.print("Loza");
            }
            if (i%7 == 0){
                System.out.print("Woza");
            }
            if (i%3 != 0 && i%5 != 0 && i%7 != 0){
                System.out.print(i);
            }
            if (i%11 == 0){
                System.out.println();
            }
            else{
                System.out.print(" ");
            }
}   }   }
```

ROLL NUMBER :

3.6 Fibonacci (Decision & Loop)

Write a program called Fibonacci to print the first 20 Fibonacci numbers F(n), where F(n) = F(n-1) + F(n-2) and F(1) = F(2) = 1. Also compute their average.

```
Public class Fibonacci {
    public static void main (String args[]){
        int n=3, fn, f1=1, f2=1, nMax = 20, sum = 0;
        System.out.println("The first " +nMax + " Fibonacci numbers
                                    are: ");
        System.out.print (0 + " " + 1 + " " + 1 + " ");

        while (n <= nMax){
            n++;
            fn = f1 + f2;
            sum += fn;
            System.out.print (fn + " ");
            f1 = f2;
            f2 = fn;
        }
        System.out.println ("\n The average is " + (double)(sum)/(nMax));
    }
}
```

Output :

```
The first 20 Fibonacci numbers are :
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987 1597 2584 4181 6765

The average is 855.4
```

ROLL NUMBER :

3.7 Extract Digits (Decision + Loop)

Write a program called ExtractDigits to extract digit from an int. In the reverse order, for example, if the int is 15423, the output shall be "3 2 4 5 1", with a space seperating the digits.

```
public class ExtractDigits {
    public static void main (String args[]){
        int n = 15423, digit;

        while (n>0){
            digit = n%10;
            n /= 10;
            System.out.print (digit + "  ");
        }
    }
}
```

output :

3 2 4 5 1