Lab #2: Recursion

The main aim of the lab is to solve some common problems using a recursive approach.

==

Task 1: Basic Problems

==

Task 1.1: Using recursive approach to implement the following **Algebra problems:**

1. $S(n)=1-2+3-4+\ldots+((-1)^{(n+1)}).n$, $n>0$

2. $S(n)=1+1.2+1.2.3+\ldots+1.2.3\ldots n$, $n>0$

3. $S(n)=1^2+2^2+3^2+....+n^2$, $n>0$

4. $S(n)=1+1/2+1/(2.4)+1/(2.4.6)+\ldots+1/(2.4.6...2n)$, $n>=0$

==

Suggestion:

```
public class Task1_1 {
// S(n)=1-2+3-4+…+ ((-1)^(n+1) ).n , n>0
     public static int getSn1(int n) {
          // TODO
          return 0;
     }
// S(n)=1+1.2+1.2.3+…+1.2.3…n, n>0
     public static int getSn2(int n) {
          // TODO
          return 0;
     }
// S(n)=1^2+2^2+3^2+....+n^2 , n>0
     public static int getSn3(int n) {
          // TODO
          return 0;
     }
// S(n)=1+1/2+1/(2.4)+1/(2.4.6)+…+1/(2.4.6.2n), n>=0
          public static double getSn4(int n) {
          // TODO
          return 0.0;
     }

}
```

==

Task 1.2: Using recursive approach to implement the **Fibonacci** problem. Note, Fibonacci – the next number is the sum of the previous two numbers.

Example: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, …

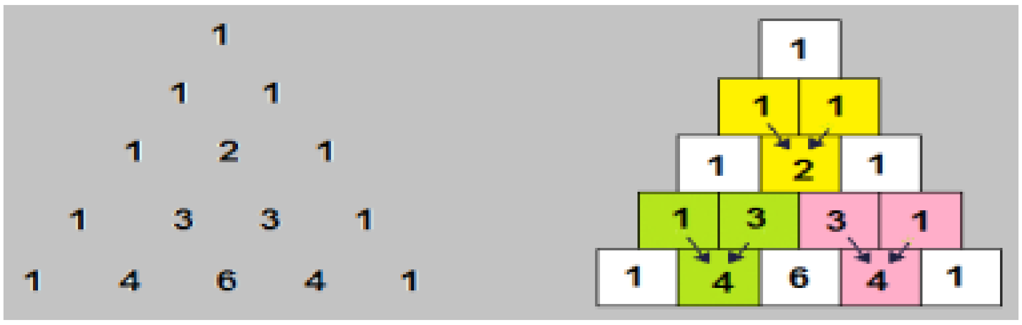
Suggestion:

```
public class Fibonacci {
     //get the nth value of the Fibonacci series
     public static int getFibonacci(int n) {
          //TODO
          return 0;
     }
// This method is used to display a Fibonaccci series based
on  the parameter n. Ex. n=10 ==> 0 1 1 2 3 5 8 13 21 34
public static void printFibonacci(int n) {
          //TODO
     }

}
```

===

Task 1.3: Using recursive approach to implement **Pascal's triangle problem:**



Suggestion:

```
public class PascalTriangle {
     // This method is used to display a Pascal triangle based
on the parameter n.
     // Where n represents the number of rows
     public static void printPascalTriangle(int row) {
          //TODO
     }

     // get the nth row.
     //For example: n=1 ==> {1}, n=2 ==> {1, 1}, ...
     public static int[] getPascalTriangle(int n) {
          //TODO
```

```
                                          }
```

```
          return null;


     // generate the next row based on the previous
row //Ex. prevRow = {1} ==> nextRow = {1, 1}
//Ex. prevRow = {1, 1} ==> nextRow = {1, 2, 1}
     public static int[] generateNextRow(int[] prevRow) {
          //TODO
          return null;
     }

}
```

Optional: How to implements these problems by using iterative approach?

===

Task 2: Advanced Problems

===

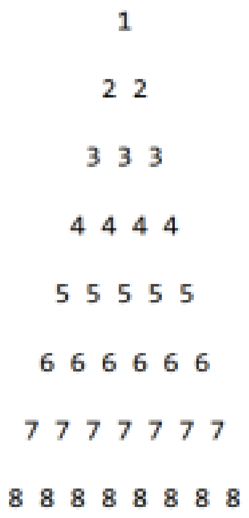
Task 2.1: Implement **drawPyramid(int n)** - This method takes as an input one integer value n and then output on console a pyramid

```
as on figure below for example for n=4:
// X
// XXX
// XXXXX
// XXXXXXX
```

```java
    public static void drawPyramid(int n) {
        //TODO
    }
```

==

Task 2.2: Using other patterns for the **drawPyramid** method defined in the previous task.



```
              1

             2 2

            3 3 3

           4 4 4 4

          5 5 5 5 5

         6 6 6 6 6 6

        7 7 7 7 7 7 7

       8 8 8 8 8 8 8 8
```