

Lab #2: Recursion

The main aim of the lab is to solve some common problems using recursion.

Task 1: Basic Problems

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

1. $S(n)=1-2+3-4+\dots+((-1)^{(n+1)}).n$, $n>0$
2. $S(n)=1+1.2+1.2.3+\dots+1.2.3\dots n$, $n>0$
3. $S(n)=1^2+2^2+3^2+\dots+n^2$, $n>0$
4. $S(n)=1+1/2+1/(2.4)+1/(2.4.6)+\dots+1/(2.4.6\dots 2n)$, $n\geq 0$

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 **Fibonacci** - next number is the sum of previous two numbers.

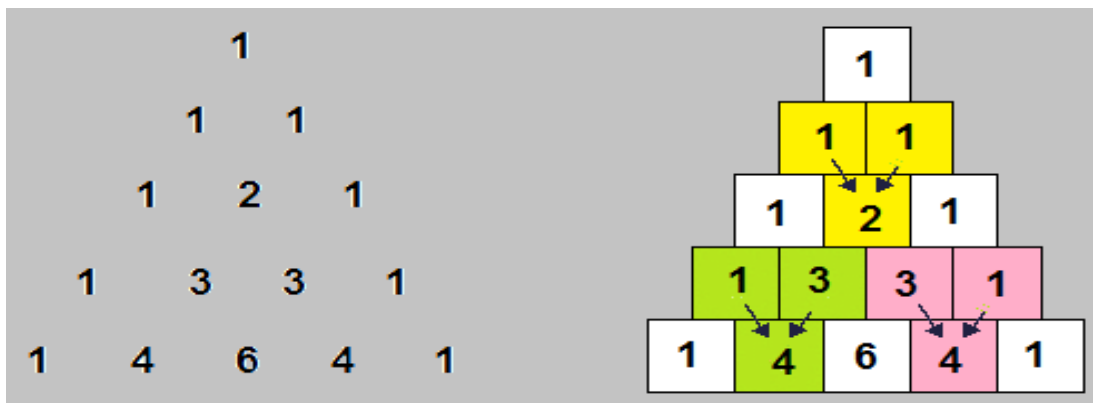
Example: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...

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 Fibonacci 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**:



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 1.4: Using recursive approach to implement **Towers of Hanoi**.

Suggestion: Use the pseudo code described in the following figure:

```
FUNCTION MoveTower(disk, source, dest, spare):  
IF disk == 0, THEN:  
    move disk from source to dest  
ELSE:  
    MoveTower(disk - 1, source, spare, dest)    // Step 1 above  
    move disk from source to dest                // Step 2 above  
    MoveTower(disk - 1, spare, dest, source)    // Step 3 above  
END IF
```

=====

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
```

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

Task 2.2 (Optional): 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
9 9 9 9 9 9 9 9 9

Pyramid Pattern-1

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

Pyramid Pattern-2

A triangular arrangement of 28 asterisks. The top row contains 10 asterisks, and each subsequent row below it contains one fewer asterisk, ending with a single asterisk in the bottom row.

Pyramid Pattern-3

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

Pyramid Pattern-4

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

Pyramid Pattern-5

Inverted Pyramid Pattern-6

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

Inverted Pyramid Pattern-7

Journal Dev

Task 2.3: Implement `drawChristmasTree(int n)` - This method takes as an input one integer value n and then output on console a Christmas tree in which last part height equals n.

The tree consists of pyramids of heights from 1 to n .

The shape have to be as presented below (for $n=4$):

//	X
//	X
//	XXX
//	X
//	XXX
//	XXXXXX
//	X
//	XXX
//	XXXXXX
//	XXXXXXXX

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