

# LAB#3

## RECURSION

**OBJECTIVE:** To understand the complexities of the recursive functions and a way to reduce these complexities.

### LABTASKS:

#### TASK#1:

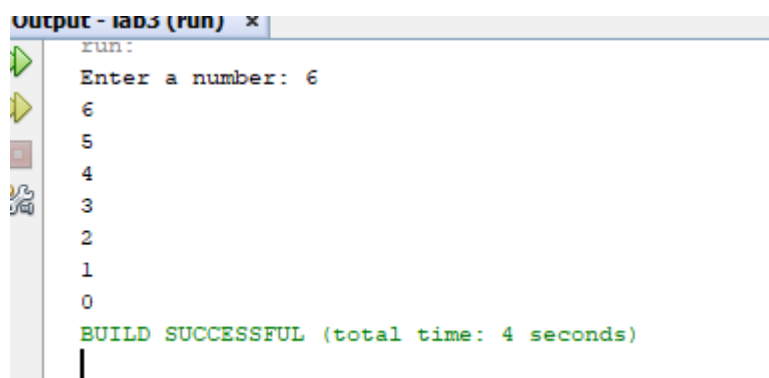
1. Write a program which takes an integer value (k) as input and prints the sequence of numbers from k to 0 in descending order

```
package lab3;
import java.util.Scanner;

public class Lab3 {
    public static void descending(int k) {
        if (k < 0) {
            return;
        }
        System.out.println(k);
        descending(k - 1);
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int k = input.nextInt();
        descending(k);
    }
}
```

#### OUTPUT:



```
Output - lab3 (run) x
run:
Enter a number: 6
6
5
4
3
2
1
0
BUILD SUCCESSFUL (total time: 4 seconds)
```

## **TASK#2:**

2. Write a program to reverse your full name using Recursion.

```
package labb3;
public class ReverseName {
    public static String reverse(String name) {
        if (name.length() == 0) {
            return name;
        }
        return name.charAt(name.length() - 1) +
            reverse(name.substring(0, name.length() - 1));
    }
    public static void main(String[] args) {
        String fullName = "Abiha omer";
        System.out.println("Reversed name: " + reverse(fullName));
    }
}
```

## **OUTPUT:**

```
run:
Reversed name: remo ahibA
BUILD SUCCESSFUL (total time: 8 seconds)
```

## **TASK#3:**

3. Write a program to calculate the sum of numbers from 1 to N using recursion. N should be user input.

```
package labb3;
import java.util.Scanner;
public class SumOfNumbers {
    public static int sum(int n) {
        if (n == 0) {
            return 0;
        }
        return n + sum(n - 1);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a positive integer N: ");
        int N = scanner.nextInt();
        int result = sum(N);
        System.out.println("The sum of numbers from 1 to " + N + " is: " + result);
    }
}
```

### Output:

```
run:
Enter a positive integer N: 8
The sum of numbers from 1 to 8 is: 36
BUILD SUCCESSFUL (total time: 4 seconds)
```

### TASK#4:

4. Write a recursive program to calculate the sum of elements in an array.

---

```
package labb3;
public class ArraySum {
    public static int sumArray(int[] array, int n) {
        if (n <= 0) {
            return 0;
        }
        return array[n - 1] + sumArray(array, n - 1);
    }
    public static void main(String[] args) {
        int[] array = {1, 2, 35, 4, 15};
        int result = sumArray(array, array.length);
        System.out.println("The sum of array elements is: " + result);
    }
}
```

### Output:

---

```
run:
The sum of array elements is: 57
BUILD SUCCESSFUL (total time: 0 seconds)
```

### Task#5:

5. Write a recursive program to calculate the factorial of a given integer n

```

2 package task2;
3 import java.util.Scanner;
4
5 public class Task2{//task5
6 public static int factorial(int n) {
7     if (n == 0) {
8         return 1;
9     }
10    return n * factorial(n - 1);
11 }
12
13 public static void main(String[] args) {
14     Scanner input= new Scanner(System.in);
15     System.out.print("Enter a integer: ");
16     int n =input.nextInt();
17     int result = factorial(n);
18     System.out.println("Factorial of " + n + " is: " + result);
19 }
20 }

```

Output - task2 (run) x

```

run:
Enter a integer: 6
Factorial of 6 is: 720
BUILD SUCCESSFUL (total time: 4 seconds)

```

## HOME TASKS:

### TASK#1:

1. Write a java program to find the N-th term in the Fibonacci series using Memoization.

```

package labb3;
import java.util.*;
public class Labb3 {
    private Map<Integer, Long> memo = new HashMap<>();

    public long fibonacci(int n) {
        if (n <= 1) return n;
        if (memo.containsKey(n)) return memo.get(n);
        long result = fibonacci(n - 1) + fibonacci(n - 2);
        memo.put(n, result);
        return result;
    }

    public static void main(String[] args) {
        Scanner input=new Scanner(System.in);
        System.out.println("enter value of n=");
        int n=input.nextInt();
    }
}

```

OUTPUT:

```
run:
enter value of n=
54
The 54-th term in the Fibonacci series is: 86267571272
BUILD SUCCESSFUL (total time: 3 seconds)
```

## **TASK#2:**

2. Write a program to count the digits of a given number using recursion.

```
1
2 package counting;
3 import java.util.Scanner;
4 public class Counting { //task6
5
6     public static int counting(int num) {
7         if (num == 0) {
8             return 0;
9         }
10        return 1 ;
11    }
12
13    public static void main(String[] args) {
14        Scanner input = new Scanner(System.in);
15        System.out.print("Enter a number: ");
16        int number = input.nextInt();
17        int totalnum = counting(Math.abs(number));
18        System.out.println("The number of digits is: " + totalnum);
19    }
20 }
```

Output - counting (run) x

```
run:
Enter a number: 2
The number of digits is: 1
BUILD SUCCESSFUL (total time: 3 seconds)
```

### **TASK#3:**

3. Write a java program to check whether a given string is a palindrome or not. A palindrome is a string that reads the same forwards and backwards. Print "YES" if the string is a palindrome, otherwise print "NO".

```
package labb3;
import java.util.Scanner;
public class palindrome {
    public static boolean isPalindrome(String str, int left, int right) {
        if (left >= right) {
            return true;
        }
        if (str.charAt(left) != str.charAt(right)) {
            return false;
        }
        return isPalindrome(str, left + 1, right - 1);
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String inp = input.nextLine().toLowerCase();
        if (isPalindrome(inp, 0, inp.length() - 1)) {
            System.out.println("YES");
        }
        else {
            System.out.println("NO");
        }
    }
}
```

### **OUTPUT:**

```
run:
Enter a string: racecar
YES
BUILD SUCCESSFUL (total time: 5 seconds)
```

### **TASK#4:**

4. Write a recursive program to find the greatest common divisor (GCD) of two numbers using Euclid's algorithm.

```
package labb3;
import java.util.Scanner;
public class gcd {
    public static int gcd(int a, int b) {
        return (b == 0) ? a : gcd(b, a % b);
    }
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Enter two numbers: ");
        int num1 = input.nextInt();
        int num2 = input.nextInt();

        int result = gcd(num1, num2);
        System.out.println("The GCD is: " + result);
    }
}
```

### **OUTPUT:**

```
run:
Enter two numbers: 1 5
The GCD is: 1
BUILD SUCCESSFUL (total time: 3 seconds)
```