LAB # 03

RECURSION

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

LAB TASK

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.

```java
import java.util.Scanner;

public class DescendingOrder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter an integer value (k): ");
        int k = scanner.nextInt();
        printDescending(k);
    }

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

Output:

```
Enter an integer value (k): 5
5
4
3
2
1
0
```

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

```java
import java.util.Scanner;

public class ReverseName {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your full name: ");
        String name = scanner.nextLine();
        String reversedName = reverseString(name);
        System.out.println("Reversed Name: " + reversedName);
    }

    public static String reverseString(String str) {
        if (str.isEmpty()) {
            return str;
        }
        return reverseString(str.substring(1)) + str.charAt(0);
    }
}
```

Output:

```
Enter your full name: John Doe
Reversed Name: eoD nhoJ
```

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

```java
import java.util.Scanner;

public class SumToN {
    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 sum = sum(n);
        System.out.println("Sum from 1 to " + n + " is: " + sum);
    }

    public static int sum(int n) {
        if (n <= 0) return 0;
        return n + sum(n - 1);
    }
}
```

Output:

```
Enter a positive integer (N): 10
Sum from 1 to 10 is: 55
```

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

```java
import java.util.Scanner;

public class SumOfArray {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int n = scanner.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = scanner.nextInt();
        }

        int sum = sumArray(arr, n);
        System.out.println("Sum of array elements is: " + sum);
    }

    public static int sumArray(int[] arr, int n) {
        if (n <= 0) return 0;
        return arr[n - 1] + sumArray(arr, n - 1);
    }
}
```

Output:

```
Enter the number of elements in the array: 5
Enter the elements of the array:
1
2
3
4
5
Sum of array elements is: 15
```

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

```java
import java.util.Scanner;

public class Factorial {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a positive integer to find its factorial: ");
        int n = scanner.nextInt();
        long factorial = factorial(n);
        System.out.println("Factorial of " + n + " is: " + factorial);
    }

    public static long factorial(int n) {
        if (n == 0) return 1;
        return n * factorial(n - 1);
    }
}
```

Output:

```
Enter a positive integer to find its factorial: 5
Factorial of 5 is: 120
```

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

```java
import java.util.Scanner;

public class CountDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        int count = countDigits(number);
        System.out.println("Number of digits: " + count);
    }

    public static int countDigits(int n) {
        if (n == 0) return 1;  // At least one digit
        return countDigitsHelper(n);
    }

    private static int countDigitsHelper(int n) {
        if (n == 0) return 0;
        return 1 + countDigitsHelper(n / 10);
    }
}
```

Output:

```
Enter a number: 12345
Number of digits: 5
```

HOME TASK

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

```java
import java.util.HashMap;
import java.util.Scanner;

public class FibonacciMemoization {
    private static HashMap<Integer, Long> memo = new HashMap<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the term number (N) in Fibonacci series: ");
        int n = scanner.nextInt();
        long fib = fibonacci(n);
        System.out.println("Fibonacci term " + n + " is: " + fib);
    }

    public static 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;
    }
```

Outpur:

```
Enter the term number (N) in Fibonacci series: 10
Fibonacci term 10 is: 55
```

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

```java
import java.util.Scanner;

public class CountDigits {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int number = scanner.nextInt();
        int count = countDigits(number);
        System.out.println("Number of digits: " + count);
    }

    public static int countDigits(int n) {
        if (n == 0) return 1;  // At least one digit
        return countDigitsHelper(n);
    }

    private static int countDigitsHelper(int n) {
        if (n == 0) return 0;
        return 1 + countDigitsHelper(n / 10);
    }
}
```

Output:

```
Enter a number: 12345
Number of digits: 5
```

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".

```java
import java.util.Scanner;

public class PalindromeCheck {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        if (isPalindrome(input)) {
            System.out.println("YES");
        } else {
            System.out.println("NO");
        }
    }

    public static boolean isPalindrome(String str) {
        return isPalindromeHelper(str, 0, str.length() - 1);
    }

    private static boolean isPalindromeHelper(String str, int left, int right) {
        if (left >= right) return true;
        if (str.charAt(left) != str.charAt(right)) return false;
        return isPalindromeHelper(str, left + 1, right - 1);
    }
}
```

Output:

```
Enter a string: racecar
YES
```

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

```java
import java.util.Scanner;

public class GCD {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the first number: ");
        int a = scanner.nextInt();
        System.out.print("Enter the second number: ");
        int b = scanner.nextInt();
        int gcd = findGCD(a, b);
        System.out.println("GCD of " + a + " and " + b + " is: " + gcd);
    }

    public static int findGCD(int a, int b) {
        if (b == 0) return a;
        return findGCD(b, a % b);
    }
}
```

Output:

```
Enter the first number: 48
Enter the second number: 18
GCD of 48 and 18 is: 6
```