LAB 03

RECURSION

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

LAB TASKS

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.**

INPUT:

```
package lab.pkg03;
import java.util.Scanner;
public class Lab03{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(source: System.in);
        System.out.print(s: "Enter an integer (k): ");
        int k = scanner.nextInt();

        for (int i = k; i >= 0; i--) {
            System.out.print(i + " ");
        }

        scanner.close();
    }
}
```

OUTPUT:

```
Enter an integer (k): 5
5 4 3 2 1 0 BUILD SUCCESSFUL (total time: 4 seconds)
```

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

INPUT:

```
import java.util.Scanner;
public class Lab03 {
    public static String reverse(String name) {
        if (name.length() <= 1) {
            return name;
        }
        return reverse(name:name.substring(beginIndex: 1)) + name.charAt(index: 0);
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(source: System.in);
        System.out.print(s: "Enter your full name: ");
        String name = sc.nextLine();
        System.out.println("Reversed Name: " + reverse(name));
        sc.close();
    }
}
```

OUTPUT:

```
Enter your full name: Laiba
Reversed Name: abiaL
```

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

INPUT:

```
import java.util.Scanner;
public class Lab03 {
    public static int calculateSum(int n) {
        if (n == 1) {
            return 1;
        }
        return n + calculateSum(n - 1);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(source: System.in);
        System.out.print(s: "Enter a number: ");
        int n = sc.nextInt();
        int sum = calculateSum(n);
        System.out.println("Sum of numbers from 1 to " + n + " is: " + sum);
        sc.close();
    }
}
```

OUTPUT:

```
Enter a number: 12
Sum of numbers from 1 to 12 is: 78
```

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

INPUT:

```java
public class Lab03 {
    public static int sumArray(int[] arr, int index) {

        if (index == arr.length) {
            return 0;
        }
        return arr[index] + sumArray(arr, index + 1);
    }
    public static void main(String[] args) {
        int[] array = {1, 2, 3, 4, 5};
        int sum = sumArray(arr: array, index: 0);

        System.out.println("Sum of elements in the array: " + sum);
    }
}
```

OUTPUT:

```
Sum of elements in the array: 15
```

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

INPUT:

```java
import java.util.Scanner;

public class Lab03 {
    public static long factorial(int n) {
        if (n == 0 || n == 1) {
            return 1;
        }
        return n * factorial(n - 1);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(source: System.in);
        System.out.print(s: "Enter a positive integer: ");

        int number = scanner.nextInt();
        if (number < 0) {
            System.out.println(x: "Factorial is not defined for negative numbers.");
        } else {
            long result = factorial(n: number);
            System.out.println("Factorial of " + number + " is: " + result);
        }
        scanner.close();
    }
}
```

OUTPUT:

```
Enter a positive integer: 5
Factorial of 5 is: 120
```

```
Enter a positive integer: -6
Factorial is not defined for negative numbers.
```

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

INPUT:

```java
import java.util.Scanner;
public class Lab03 {
    public static int countDigits(int number) {
        if (number == 0) {
            return 0;
        }
        return 1 + countDigits(number / 10);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(source: System.in);
        System.out.print(s: "Enter a positive integer: ");

        int number = scanner.nextInt();
        if (number == 0) {
            System.out.println(x: "The number of digits is: 1");
        } else if (number < 0) {
            System.out.println(x: "Please enter a positive integer.");
        } else {
            int digitCount = countDigits(number);
            System.out.println("The number of digits is: " + digitCount);
        }

        scanner.close();
    }
}
```

OUTPUT:

```
Enter a positive integer: 780
The number of digits is: 3
```

HOME TASK

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

INPUT:

```
import java.util.Scanner;
public class Lab03 {
    public static long fibonacci(int n, long[] memo) {
        if (n == 0) {
            return 0;
        }
        if (n == 1) {
            return 1;
        }
        if (memo[n] != -1) {
            return memo[n];
        }
        memo[n] = fibonacci(n - 1, memo) + fibonacci(n - 2, memo);
        return memo[n];
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(source: System.in);
        System.out.print(s: "Enter the term number (N) for Fibonacci series: ");
        int n = scanner.nextInt();
        if (n < 0) {
            System.out.println(x: "Please enter a non-negative integer.");
        } else {
            long[] memo = new long[n + 1];
            for (int i = 0; i < memo.length; i++) {
                memo[i] = -1;
            }
            long nthFibonacci = fibonacci(n, memo);
            System.out.println("The " + n + "-th term in the Fibonacci series is: " + nthFibonacci);
        }
        scanner.close();
```

OUTPUT:

```
Enter the term number (N) for Fibonacci series: 10
The 10-th term in the Fibonacci series is: 55
```

2. **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".**

INPUT:

```
import java.util.Scanner;
public class Lab03 {
    public static boolean isPalindrome(String str) {
        int left = 0;
        int right = str.length() - 1;

        while (left < right) {
            if (str.charAt(index: left) != str.charAt(index: right)) {
                return false;
            }
            left++;
            right--;
        }
        return true;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(source: System.in);
        System.out.print(s: "Enter a string: ");

        String input = scanner.nextLine();
        String sanitizedInput = input.replaceAll(regex: "\\s+", replacement: "").toLowerCase();

        if (isPalindrome(str: sanitizedInput)) {
            System.out.println(x: "YES");
        } else {
            System.out.println(x: "NO");
        }
        scanner.close();
    }
}
```

OUTPUT:

```
Enter a string: Maham
YES
```

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

INPUT:

```
import java.util.Scanner;
public class Lab03 {
    public static int gcd(int a, int b) {
        if (b == 0) {
            return a;
        }
        return gcd(a: b, a % b);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(source: System.in);
        System.out.print(s: "Enter the first number (a): ");
        int a = scanner.nextInt();
        System.out.print(s: "Enter the second number (b): ");
        int b = scanner.nextInt();

        if (a < 0 || b < 0) {
            System.out.println(x: "Please enter non-negative integers.");
        } else {
            int result = gcd(a, b);
            System.out.println("The GCD of " + a + " and " + b + " is: " + result);
        }
        scanner.close();
    }
}
```

OUTPUT:

```
Enter the first number (a): 12
Enter the second number (b): 24
The GCD of 12 and 24 is: 12
```