LAB NO 3

RECURSION

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

# Question 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:

import java.util.Scanner;

public class DescendingSequence {

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 n) {

if (n < 0) {

return;

}

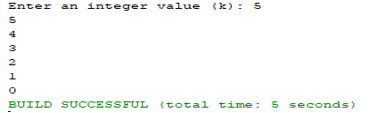
System.out.print(n + " ");

printDescending(n - 1);

}

}

**OUTPUT:**

****

# Question 2:

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

. package lab03;

import java.util.Scanner;

public class SumRecurrsion {

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);

scanner.close();

}

}

**OUTPUT:**

****

# Question 3:

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

# INPUT:

public class ArraySum {

public static void main(String[] args) {

int[] array = {1, 2, 3, 4, 5};

int sum = calculateSum(array, array.length);

System.out.println("Sum of array elements: " + sum);

}

public static int calculateSum(int[] array, int n) {

if (n <= 0) {

return 0;

}

return calculateSum(array, n - 1) + array[n - 1];

}  
**OUTPUT:**



# Question 4:

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

**INPUT:**

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: ");

int n = scanner.nextInt();

System.out.println("Factorial of " + n + " is: " + factorial(n));

scanner.close();

}

public static int factorial(int n) {

if (n == 0) {

return 1;

} else {

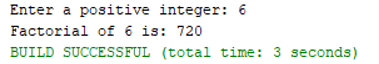
return n \* factorial(n - 1);

}

}

}

**OUTPUT:**

****

# Question 5:

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

# INPUT:

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 number) {

if (number < 10) {

return 1;

}

return 1 + countDigits(number / 10);

}

}

**OUTPUT:**

****

# Question 1:

**HOME TASKS**

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

# INPUT:

import java.util.HashMap;

public class Fibonacci {

private static HashMap<Integer, Long> memo = new HashMap<>();

public static void main(String[] args) {

int n = 10;

System.out.println("The " + n + "-th term in the Fibonacci series is: " + fibonacci(n));

}

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;

}

}

**OUTPUT:**

****

# Question 2:

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

# INPUT:

import java.util.Scanner;

public class countsdigits {

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 number) {

if (number < 10) {

return 1;

} else {

return 1 + countDigits(number / 10);

}

}

}

**OUTPUT:**

****

# Question 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".

# INPUT:

import java.util.Scanner;

public class PalindromeChecker {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = scanner.nextLine();

String reversed = new StringBuilder(input).reverse().toString();

if (input.equals(reversed)) {

System.out.println("YES");

} else {

System.out.println("NO");

}

scanner.close();

}

}

**OUTPUT:**

****

# Question 4:

# 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 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("The GCD of " + a + " and " + b + " is: " + gcd);

}

public static int findGCD(int a, int b) {

if (b == 0) {

return a;

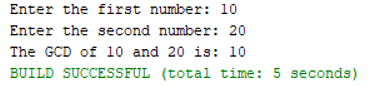
}

return findGCD(b, a % b);

}

}

**OUTPUT:**

****