# LAB # 03

**LAB TASKS 1:**

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();

System.out.println("Sequence from " + k + " to 0 in descending order:");

for (int i = k; i >= 0; i--) {

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

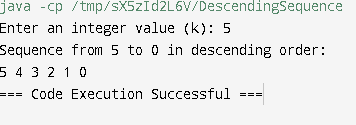
}

scanner.close();

}

}

**OUTPUT**

****

**LAB TASKS 2:**

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();

System.out.println("Reversed name: " + reverseString(name));

scanner.close();

}

public static String reverseString(String str) {

if (str.isEmpty()) {

return str;

}

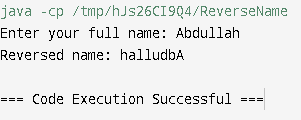
// Recursive case: get the last character + reverse the rest of the string

return reverseString(str.substring(1)) + str.charAt(0);

}

}

**OUTPUT**

****

**LAB TASKS 3:**

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 = calculateSum(N);

System.out.println("Sum of numbers from 1 to " + N + " is: " + sum);

scanner.close();

}

public static int calculateSum(int n) {

// Base case: if n is 1, return 1

if (n == 1) {

return 1;

}

// Recursive case: sum of n + sum of numbers from 1 to (n-1)

return n + calculateSum(n - 1);

}

}

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**LAB TASKS 4:**

public class ArraySum {

public static void main(String[] args) {

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

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

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

}

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

// Base case: If the array is empty (n is 0), the sum is 0

if (n <= 0) {

return 0;

}

// Recursive case: Sum of last element + sum of the rest of the array

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

}

}

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**LAB TASKS 5:**

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 (n): ");

int n = scanner.nextInt();

int result = factorial(n);

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

scanner.close();

}

public static int factorial(int n) {

// Base case: if n is 0 or 1, the factorial is 1

if (n <= 1) {

return 1;

}

// Recursive case: n \* factorial of (n-1)

return n \* factorial(n - 1);

}

}

**OUTPUT**

**A screenshot of a computer

Description automatically generated**

**LAB TASKS 6:**

import java.util.Scanner;

public class DigitCounter {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

int number = scanner.nextInt();

int digitCount = countDigits(number);

System.out.println("Number of digits: " + digitCount);

scanner.close();

}

public static int countDigits(int n) {

// Base case: if n is 0, it means we've processed all digits

if (n == 0) {

return 0;

}

// Recursive case: 1 (for the current digit) + count of the remaining digits

return 1 + countDigits(n / 10);

}

}

**OUTPUT**

A screenshot of a computer program

Description automatically generated

**HOME TASK 1:**

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 (N) to find in the Fibonacci series: ");

int n = scanner.nextInt();

long nthTerm = fibonacci(n);

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

scanner.close();

}

public static long fibonacci(int n) {

// Base cases

if (n <= 1) {

return n;

}

// Check if the value is already computed and stored in the memo map

if (memo.containsKey(n)) {

return memo.get(n);

}

// Recursive case with memoization

long result = fibonacci(n - 1) + fibonacci(n - 2);

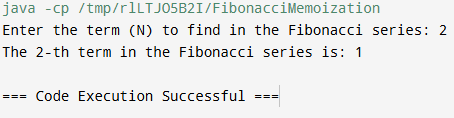
memo.put(n, result); // Store the computed result for future use

return result;

}

}

**OUTPUT**



**HOME TASK 2:**

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");

}

scanner.close();

}

public static boolean isPalindrome(String str) {

int left = 0;

int right = str.length() - 1;

// Check characters from both ends towards the middle

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false; // Not a palindrome

}

left++;

right--;

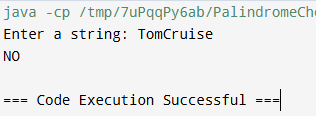
}

return true; // It's a palindrome

}

}

**OUTPUT**



**HOME TASK 3:**

import java.util.Scanner;

public class GCDRecursive {

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);

scanner.close();

}

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

// Base case: if b is 0, then gcd is a

if (b == 0) {

return a;

}

// Recursive case: call findGCD with (b, a % b)

return findGCD(b, a % b);

}

}

**OUTPUT**

