Kashaf Khan 2023F-BSE-248 Section: A

LAB # 03

Recursion

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

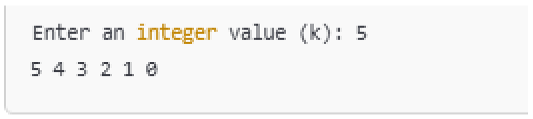
LAB TASKS:

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

CODE:

```
public class Sequence {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter an integer value (k): ");
    int k = scanner.nextInt();
    printSequence(k);}
  public static void printSequence(int k) {
    if (k < 0) {
      return; // Base case: if k is less than 0, stop recursion   }
    System.out.print(k + " ");
    printSequence(k - 1); // Recursive call with k - 1
  }
}
```

OUTPUT



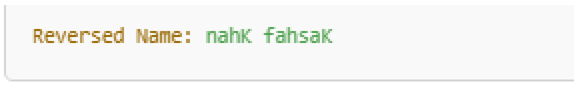
```
Enter an integer value (k): 5
5 4 3 2 1 0
```

Write a program to reverse your full name using Recursion.

CODE:

```
public class ReverseName {
   public static void main(String[] args) {
      String name = "Kashaf Khan";
      System.out.print("Reversed Name: ");
      reverseName(name, name.length() - 1);    }
public static void reverseName(String name, int index) {
      if (index < 0) {
         return; // Base case: if index is less than 0, stop recursion
      System.out.print(name.charAt(index)); // Print character at current index
      reverseName(name, index - 1); // Recursive call with index - 1  }
}
```

OUTPUT:



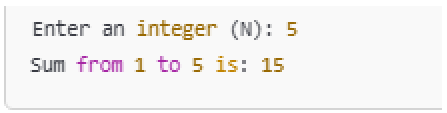
```
Reversed Name: nahK fahsaK
```

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

CODE:

```
public class SumToN {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter an integer (N): ");
      int N = scanner.nextInt()
      System.out.println("Sum from 1 to " + N + " is: " + sumToN(N)); }
   public static int sumToN(int N) {
      if (N <= 1) {
         return N; // Base case: if N is 1 or less, return N   }
      return N + sumToN(N - 1); // Recursive call with N -     }
}
```

OUTPUT:



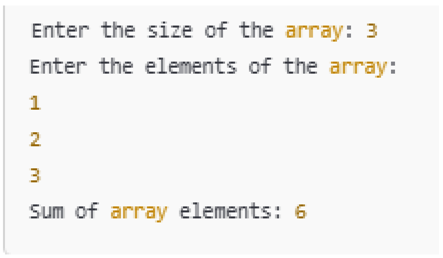
```
Enter an integer (N): 5
Sum from 1 to 5 is: 15
```

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

CODE:

```
public class ArraySum {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter the size of the array: ");
    int size = scanner.nextInt();
    int[] array = new int[size];
    System.out.println("Enter the elements of the array:");
    for (int i = 0; i < size; i++) {
      array[i] = scanner.nextInt();
    }
    System.out.println("Sum of array elements: " + sumArray(array, size - 1));
  }
  public static int sumArray(int[] array, int index) {
    if (index < 0) {
      return 0; // Base case: if index is less than 0, return 0
    return array[index] + sumArray(array, index - 1); // Recursive call with index – 1
```

OUTPUT:



```
Enter the size of the array: 3
Enter the elements of the array:
1
2
3
Sum of array elements: 6
```

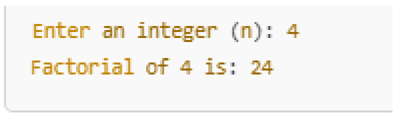
5 Write a recursive program to calculate the factorial of a given integer n

CODE:

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

```
public class Factorial {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter an integer (n): ");
      int n = scanner.nextInt();
      System.out.println("Factorial of " + n + " is: " + factorial(n));
   }
   public static int factorial(int n) {
      if (n <= 1) {
         return 1; // Base case: if n is 1 or less, return 1
      }
      return n * factorial(n - 1); // Recursive call with n - 1
   }
}
```

OUTPUT:



```
Enter an integer (n): 4
Factorial of 4 is: 24
```

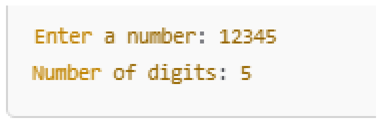
6 Write a program to count the digits of a given number using recursion.

CODE:

```
import java.util.Scanner;
public class DigitCount {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      System.out.print("Enter a number: ");
      int number = scanner.nextInt();
      System.out.println("Number of digits: " + countDigits(number));
      public static int countDigits(int number) {
      if (number == 0) {
         return 0; // Base case: if number is 0, return 0
```

```
        }
        return 1 + countDigits(number / 10); // Recursive call with number divided by 10
```

OUTPUT:



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

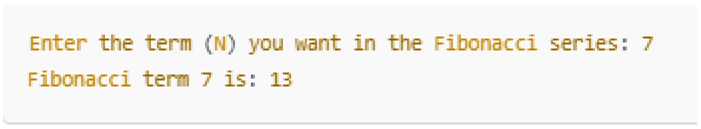
HOME TASK:

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

CODE:

```
public class FibonacciMemoization {
    private static HashMap<Integer, Integer> memo = new HashMap<>(); // Memoization map
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the term (N) you want in the Fibonacci series: ");
        int N = scanner.nextInt();
        System.out.println("Fibonacci term " + N + " is: " + fibonacci(N));    }
    public static int fibonacci(int N) {
        if (N <= 1) {
            return N; // Base case: Fibonacci(0) = 0, Fibonacci(1) = 1
        }
        if (memo.containsKey(N)) {
            return memo.get(N); // Return cached result if available
        }
        int result = fibonacci(N - 1) + fibonacci(N - 2); // Recursive calls
        memo.put(N, result); // Store the result in memo
        return result;
    }
}
```

OUTPUT:



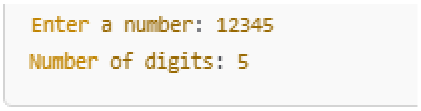
```
Enter the term (N) you want in the Fibonacci series: 7
Fibonacci term 7 is: 13
```

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

CODE:

```
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();
    System.out.println("Number of digits: " + countDigits(number));
  }
  public static int countDigits(int number) {
    if (number == 0) {
      return 0; // Base case: if number is 0, return 0
    }
    return 1 + countDigits(number / 10); // Recursive call with number divided by 10
  }
}
```

OUTPUT:



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

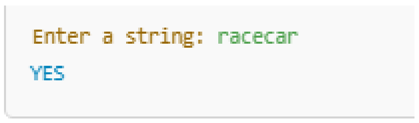
Write a java program to check whether a given string is a palindrome or not. A palindrome is a string that reads the same forwards and backwards.Print "YES" if the string is a palindrome, otherwise print "NO".

CODE:

```
public class PalindromeChecker {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
```

```
        System.out.print("Enter a string: ");
        String str = scanner.nextLine();
        if (isPalindrome(str, 0, str.length() - 1)) {
            System.out.println("YES");
        } else {
            System.out.println("NO");
        }
    public static boolean isPalindrome(String str, int left, int right) {
        if (left >= right) {
            return true; // Base case: if pointers have crossed, it is a palindrome
        }
        if (str.charAt(left) != str.charAt(right)) {
            return false; // If characters at pointers are different, not a palindrome
        }
        return isPalindrome(str, left + 1, right - 1); // Recursive call with updated pointers
 }
```

OUTPUT:



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

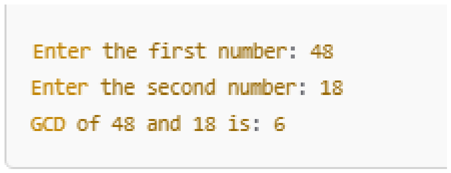
Write a recursive program to find the greatest common divisor (GCD) of two numbers using Euclid's algorithm.

CODE:

```
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();
        System.out.println("GCD of " + a + " and " + b + " is: " + gcd(a, b));

    public static int gcd(int a, int b) {
        if (b == 0) {
            return a; // Base case: if b is 0, GCD is a}
        return gcd(b, a % b); // Recursive call with b and a % b
```

OUTPUT:



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