ST. ANDREWS INSTITUTE

OF TECHNOLOGY & MANAGEMENT

Gurugram Delhi (NCR)
Approved by AICTE, Govt. of India, New Delhi
Affiliated to Maharshi Dayanand University
'A+' Grade State University, accredited by NAAC

Session: 2024 – 2025

Practical Software Lab:-Based on Advance Data Structure Using Java/C++

Course Code: 20MCA21CL5

Semester: 1st

Master of Computer Applications

Submitted By
Name: Paras Agarwal
Roll No.:24MCA061

DEPARTMENT OF COMPUTER APPLICATIONS

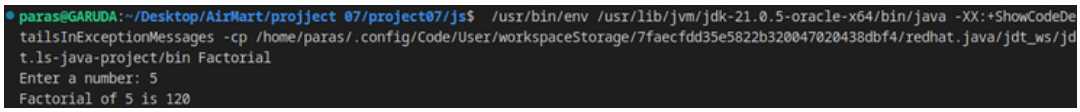
PROGRAM 1- WAP TO CALCULATE THE FACTORIAL OF A GIVEN NUMBER USING RECURSION

```
import java.util.Scanner;

public class FactorialCalculator {
    private static int factorial(int n) {
        return n == 0 ? 1 : n * factorial(n - 1);
    }

    public static void main(String[] args) {
        System.out.print("Enter a number: ");
        int number = new Scanner(System.in).nextInt();
        System.out.println("Factorial of " + number + " is " + factorial(number));
    }
}
```

OUTPUT



PROGRAM 2 - IMPLEMENT A RECURSIVE FUNCTION TO GENERATE TO FIBONACCI SEQUENCE.

```
iimport java.util.Scanner;

public class FibonacciSequence {
 private static int fibonacci(int n) {
return n <= 1 ? n : fibonacci(n - 1) + fibonacci(n - 2);
 }

 public static void main(String[] args) {
 System.out.print("Enter the number of terms: ");
 int terms = new Scanner(System.in).nextInt();
 System.out.print("Fibonacci Sequence: ");
 for (int i = 0; i < terms; i++) {
 System.out.print(fibonacci(i) + " ");
 }
 }
}
```

OUTPUT

```
Enter the number of terms: 9
Fibonacci Sequence: 0 1 1 2 3 5 8 13 21
```

PROGRAM 3 - SOLVE THE TOWER OF HANOI PUZZLE USING RECURSION.

```
import java.util.Scanner;

public class TowerOfHanoi {
   private static void solveHanoi(int n, char from, char to, char aux) {
      if (n == 0) return;
      solveHanoi(n - 1, from, aux, to);
      System.out.println("Move disk " + n + " from " + from + " to " + to);
      solveHanoi(n - 1, aux, to, from);
   }

   public static void main(String[] args) {
      System.out.print("Enter the number of disks: ");
      int disks = new Scanner(System.in).nextInt();
      solveHanoi(disks, 'A', 'C', 'B');
   }
}
```

OUTPUT

```
Enter the number of disks: 3
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

PROGRAM 4 - IMPLEMENT A RECURSIVE BINARY SEARCH ALGORITHM TO FIND AN ELEMENT IN A SORTED ARRAY.

```
import java.util.Scanner;

public class BinarySearch {
    private static int binarySearch(int[] arr, int low, int high, int target) {
        if (low > high) return -1;
        int mid = low + (high - low) / 2;
        if (arr[mid] == target) return mid;
        return arr[mid] > target ? binarySearch(arr, low, mid - 1, target) : binarySearch(arr, mid + 1, high, target);
    }

    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[] arr = new int[size];
        System.out.println("Enter the sorted array elements:");
        for (int i = 0; i < size; i++) arr[i] = scanner.nextInt();
        System.out.print("Enter the target element to search: ");
        int target = scanner.nextInt();
        int result = binarySearch(arr, 0, size - 1, target);
        System.out.println(result == -1 ? "Element not found." : "Element found at index " + result + ".");
    }
}
```

OUTPUT

```
Enter the size of the array: 4
Enter the sorted array elements:23 25 27 28
Enter the target element to search: 27
Element found at index 2.
```

PROGRAM 5 - IMPLEMENT A RECURSIVE FUNCTION TO CALCULATE THE POWER OF A NUMBER.

```
import java.util.Scanner;

public class PowerCalculator {
    private static int power(int base, int exponent) {
        return exponent == 0 ? 1 : base * power(base, exponent - 1);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the base: ");
        int base = scanner.nextInt();
        System.out.print("Enter the exponent: ");
        int exponent = scanner.nextInt();
        System.out.println(base + " raised to the power " + exponent + " is " + power(base, exponent));
    }
}
```

OUTPUT

```
Enter the base: 5
Enter the exponent: 2
5 raised to the power 2 is 25
```

PROGRAM 6 - IMPLEMENT THE MERGE SORT ALGORITHM TO SORT AN ARRAY USING THE DIVIDE AND CONQUER APPROACH.

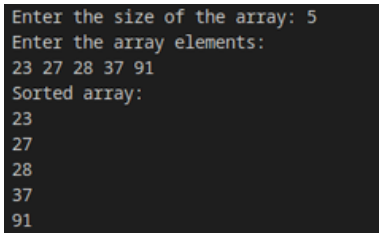
```
import java.util.Scanner;

public class MergeSort {
   private static void merge(int[] arr, int left, int mid, int right) {
      int n1 = mid - left + 1, n2 = right - mid;
      int[] L = new int[n1], R = new int[n2];
      System.arraycopy(arr, left, L, 0, n1);
      System.arraycopy(arr, mid + 1, R, 0, n2);
      int i = 0, j = 0, k = left;
      while (i < n1 && j < n2) arr[k++] = L[i] <= R[j] ? L[i++] : R[j++];
      while (i < n1) arr[k++] = L[i++];
      while (j < n2) arr[k++] = R[j++];
   }

   private static void mergeSort(int[] arr, int left, int right) {
      if (left < right) {
         int mid = left + (right - left) / 2;
         mergeSort(arr, left, mid);
         mergeSort(arr, mid + 1, right);
         merge(arr, left, mid, right);
      }
   }

   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[] arr = new int[size];
      System.out.println("Enter the array elements:");
      for (int i = 0; i < size; i++) arr[i] = scanner.nextInt();
      mergeSort(arr, 0, size - 1);
      System.out.println("Sorted array:");
      for (int num : arr) System.out.print(num + " ");
   }
}
```

OUTPUT



PROGRAM 7 -IMPLEMENT THEW QUICK SORT ALGORITHM TO SORT AN ARRAY USING THE DIVIDE AND CONQUER APPROACH.

```
import java.util.Scanner;

public class QuickSort {
   private static void quickSort(int[] arr, int low, int high) {
      if (low < high) {
         int pi = partition(arr, low, high);
         quickSort(arr, low, pi - 1);
         quickSort(arr, pi + 1, high);
      }
   }

   private static int partition(int[] arr, int low, int high) {
      int pivot = arr[high], i = low - 1;
      for (int j = low; j < high; j++) {
         if (arr[j] < pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
         }
      }
      int temp = arr[i + 1];
      arr[i + 1] = arr[high];
      arr[high] = temp;
      return i + 1;
   }

   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[] arr = new int[size];

}
```

```
System.out.println("Enter the array elements:");
for (int i = 0; i < size; i++) arr[i] = scanner.nextInt();
quickSort(arr, 0, size - 1);
System.out.println("Sorted array:");
for (int num : arr) System.out.println(num + " ");
}
```

OUTPUT