TASK 3: ARRAY MANIPULATION AND ANALYSIS PROGRAMS

(A) RETURN THE MAX SLIDING WINDOW

AIM

To write a Java program to find the maximum value in each sliding window of size k as the window moves from left to right across a given integer array.

ALGORITHM

1. Start
2. Read the array nums and window size k
3. Create an output array of size n - k + 1
4. For each window starting from index i = 0 to n - k:
    - Assume the first element of the window is the maximum
    - Compare all k elements in the current window
    - Update the maximum value
    - Store the maximum in the result array
5. Print the result

PROGRAM

```
import java.util.Scanner;

class SlidingWindowMaximum {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      // Read array size
      System.out.print("Enter array size: ");
      int n = sc.nextInt();
      int[] nums = new int[n];

      // Read array elements
      System.out.println("Enter " + n + " array elements:");
      for (int i = 0; i < n; i++) {
         nums[i] = sc.nextInt();
      }
```

```
        // Read window size
        System.out.print("Enter window size: ");
        int k = sc.nextInt();

        int[] result = new int[n - k + 1];

        // Sliding window logic
        for (int i = 0; i <= n - k; i++) {
            int max = nums[i];
            for (int j = i; j < i + k; j++) {
                if (nums[j] > max) {
                    max = nums[j];
                }
            }
            result[i] = max;
        }

        // Print result
        System.out.print("Sliding window maximum: ");
        for (int i = 0; i < result.length; i++) {
            System.out.print(result[i] + " ");
        }
    }
}
```

OUTPUT

Enter array size: 8

Enter 8 array elements:

1 3 -1 -3 5 3 6 7

Enter window size: 3

Sliding window maximum: 3 3 5 5 6 7

RESULT

The Java program successfully finds the maximum element in each sliding window of size k. The output correctly represents the maximum values for all window positions as the window moves from left to right.

(B) ADJUSTED SUM

AIM

To write a Java program that calculates the adjusted sum of an integer array by adding all even numbers and subtracting all odd numbers.

ALGORITHM

1. Start the program.
2. Read the value of N (size of the array).
3. Read N integer elements into the array.
4. Initialize a variable sum to 0.
5. Traverse each element of the array:
    - If the element is even, add it to sum.
    - If the element is odd, subtract it from sum.
6. Display the final adjusted sum.
7. Stop the program.

PROGRAM

```
import java.util.Scanner;

class AdjustedSum {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the size of the array: ");
      int N = sc.nextInt();

      int[] arr = new int[N];
      int sum = 0;

      System.out.println("Enter the array elements:");
      for (int i = 0; i < N; i++) {
         arr[i] = sc.nextInt();

         if (arr[i] % 2 == 0) {
            sum = sum + arr[i];   // Add even number
         } else {
            sum = sum - arr[i];   // Subtract odd number
```

```
            }
        }

        System.out.println("Adjusted Sum = " + sum);
    }
}
```

OUTPUT

Enter the size of the array: 5

Enter the array elements:

10 15 20 25 30

Adjusted Sum = 20

RESULT

Thus, the Java program successfully calculates the adjusted sum of the array by adding even elements and subtracting odd elements.

(C) ROTATE ARRAY AND FIND MAX DIFFERENCE

AIM

To write a Java program that rotates an array to the right by K positions and finds the maximum absolute difference between any two adjacent elements after rotation.

ALGORITHM

1. Start the program.
2. Read the size of the array N.
3. Read N elements into the array.
4. Read the value of K.
5. Rotate the array to the right by K positions using the formula: rotated[(i + K) % N] = arr[i].
6. Traverse the rotated array and calculate the absolute difference between adjacent elements using Math.abs().
7. Store the maximum difference found.
8. Display the rotated array and the maximum difference.
9. Stop the program.

PROGRAM

```
import java.util.Scanner;

class RotateMaxDifference {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the size of the array: ");
      int N = sc.nextInt();

      int[] arr = new int[N];
      int[] rotated = new int[N];

      System.out.println("Enter the array elements:");
      for (int i = 0; i < N; i++) {
         arr[i] = sc.nextInt();
      }

      System.out.print("Enter K value: ");
```

```
        int K = sc.nextInt();

        K = K % N; // Handle cases where K > N

        // Rotate array to the right
        for (int i = 0; i < N; i++) {
            rotated[(i + K) % N] = arr[i];
        }

        // Find maximum absolute difference
        int maxDiff = 0;
        for (int i = 0; i < N - 1; i++) {
            int diff = Math.abs(rotated[i] - rotated[i + 1]);
            if (diff > maxDiff) {
                maxDiff = diff;
            }
        }

        // Display rotated array
        System.out.println("Rotated Array:");
        for (int num : rotated) {
            System.out.print(num + " ");
        }

        System.out.println("\nMaximum difference = " + maxDiff);
    }
}
```

OUTPUT

Enter the size of the array: 5

Enter the array elements:

3 8 9 7 6

Enter K value: 1

Rotated Array:

6 3 8 9 7

Maximum difference = 5

RESULT

Thus, the Java program successfully rotates the array to the right by K positions and finds the maximum absolute difference between adjacent elements.