ASSIGNMENT OF DATA STRUCTURES AND ALGORITHMS

NAME: SAFEER UDDIN

ROLL NUMBER: 2K24/CSE/162

GROUP: PRE-ENGINEERING

TEACHER: DR GULSHER LAGHARI

TASK 1: Write a program to reverse an array using stack data structure.

CODE:

```
import java.util.Stack;

public class ReverseArrayUsingStack {
  public static int[] reverseArray(int[] arr) {
    // Check for null or empty array
    if (arr == null || arr.length == 0) {
      return arr; // Return the same array if it's null or empty
    }

    Stack<Integer> stack = new Stack<>();

    // Push elements into the stack
    for (int num : arr) {
      stack.push(num);
    }

    // Pop elements from the stack to reverse the array
    int[] reversedArr = new int[arr.length];
    for (int i = 0; i < arr.length; i++) {
      reversedArr[i] = stack.pop();
    }

    return reversedArr;
  }

  public static void main(String[] args) {
    int[] arr = {1, 2, 3, 4, 5};
    int[] reversedArr = reverseArray(arr);

    System.out.print("Original Array: ");
    for (int num : arr) {
      System.out.print(num + " ");
```

```
    }

    System.out.print("\nReversed Array: ");
    for (int num : reversedArr) {
      System.out.print(num + " ");
    }
  }
}
```

OUTPUT:

```
Original Array: 1 2 3 4 5
Reversed Array: 5 4 3 2 1
```

TASK 2: Write a java program to match the parentheses stored in a string using stack data structure.

CODE:

```
import java.util.Stack;
public class ParenthesesMatching {

  public static boolean isBalanced(String expression) {
    Stack<Character> stack = new Stack<>();

    for (char ch : expression.toCharArray()) {
      // Use switch-case for better readability
      switch (ch) {
        case '(': case '{': case '[':
          stack.push(ch);
          break;
        case ')':
        case '}':
```

```
            case ']':
              if (stack.isEmpty() || !isMatchingPair(stack.pop(), ch)) {
                return false;
              }
              break;
            default:
              // Ignore non-parenthesis characters or handle them if necessary
              break;
          }
        }

        return stack.isEmpty();
    }

    private static boolean isMatchingPair(char open, char close) {
        return (open == '(' && close == ')') ||
               (open == '{' && close == '}') ||
               (open == '[' && close == ']');
    }

    public static void main(String[] args) {
        String expression = "{[()()]}";
        System.out.println("Expression: " + expression);
        System.out.println("Balanced: " + isBalanced(expression));
    }
}
```

OUTPUT:

```
Expression: {[()()]}
Balanced: true
```

TASK 3: Write a java program to calculate the sum of all integer elements in an integer array by implementing a recursive sum method/ function.

CODE:

```
import java.util.Arrays;

public class RecursiveSum {

  public static int recursiveSum(int[] arr, int index) {
    // Validate the input
    if (arr == null) {
      throw new IllegalArgumentException("Array cannot be null");
    }
    // Base case: if index is equal to the length of the array, return 0
    if (index >= arr.length) {
      return 0;
    }
    // Recursive case: sum the current element and the sum of the rest
    return arr[index] + recursiveSum(arr, index + 1);
  }

  public static void main(String[] args) {
    int[] arr = {1, 2, 3, 4, 5};
    int sum = recursiveSum(arr, 0);

    // Using Arrays.toString for cleaner output
    System.out.println("Array: " + Arrays.toString(arr));
    System.out.println("Sum of Elements: " + sum);
  }
}
```

OUTPUT:

```
Array: 1 2 3 4 5
```

Sum of Elements: 15