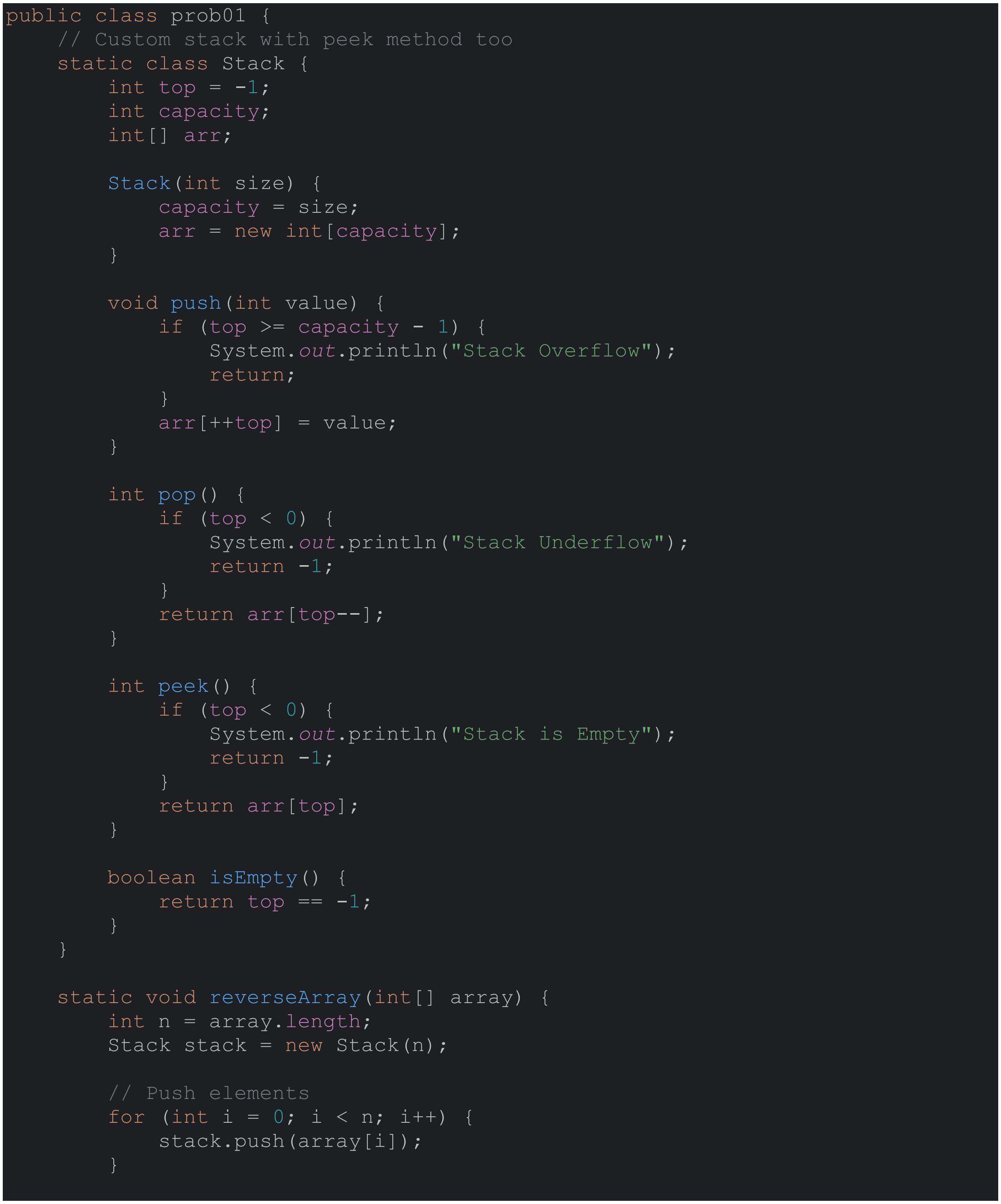
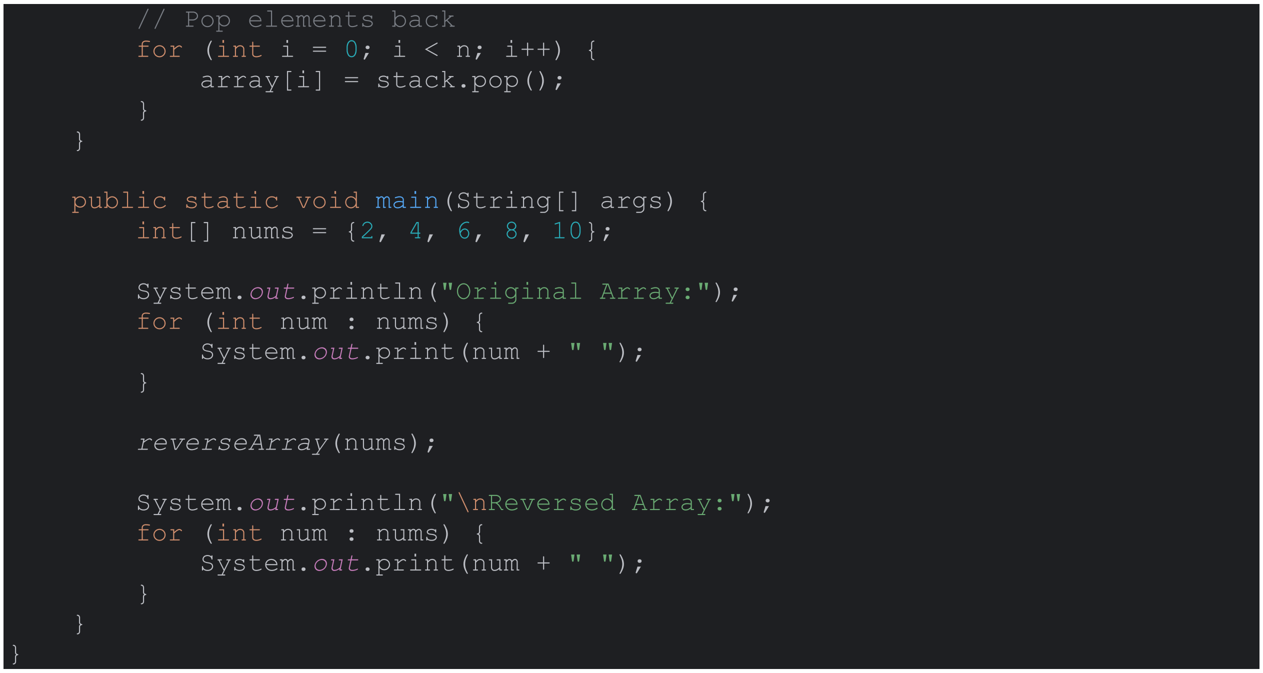
PROGRAM:1

WRITE A PROGRAM TO REVERSE AN ARRAY USING STACK DATA STRUCTURE.



```
public class prob01 {
    // Custom stack with peek method too
    static class Stack {
        int top = -1;
        int capacity;
        int[] arr;

        Stack(int size) {
            capacity = size;
            arr = new int[capacity];
        }

        void push(int value) {
            if (top >= capacity - 1) {
                System.out.println("Stack Overflow");
                return;
            }
            arr[++top] = value;
        }

        int pop() {
            if (top < 0) {
                System.out.println("Stack Underflow");
                return -1;
            }
            return arr[top--];
        }

        int peek() {
            if (top < 0) {
                System.out.println("Stack is Empty");
                return -1;
            }
            return arr[top];
        }

        boolean isEmpty() {
            return top == -1;
        }
    }

    static void reverseArray(int[] array) {
        int n = array.length;
        Stack stack = new Stack(n);

        // Push elements
        for (int i = 0; i < n; i++) {
            stack.push(array[i]);
        }
```



```
        // Pop elements back
        for (int i = 0; i < n; i++) {
            array[i] = stack.pop();
        }
    }

    public static void main(String[] args) {
        int[] nums = {2, 4, 6, 8, 10};

        System.out.println("Original Array:");
        for (int num : nums) {
            System.out.print(num + " ");
        }

        reverseArray(nums);

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

OUTPUT;

```
Original Array:
2 4 6 8 10

Reversed Array:
10 8 6 4 2
```

PROGRAM:2
TO WRITE A PROGRAM TO MATCH THE PARENTHESES STORED IN A STRING USING STACK DATA STRUCTURE

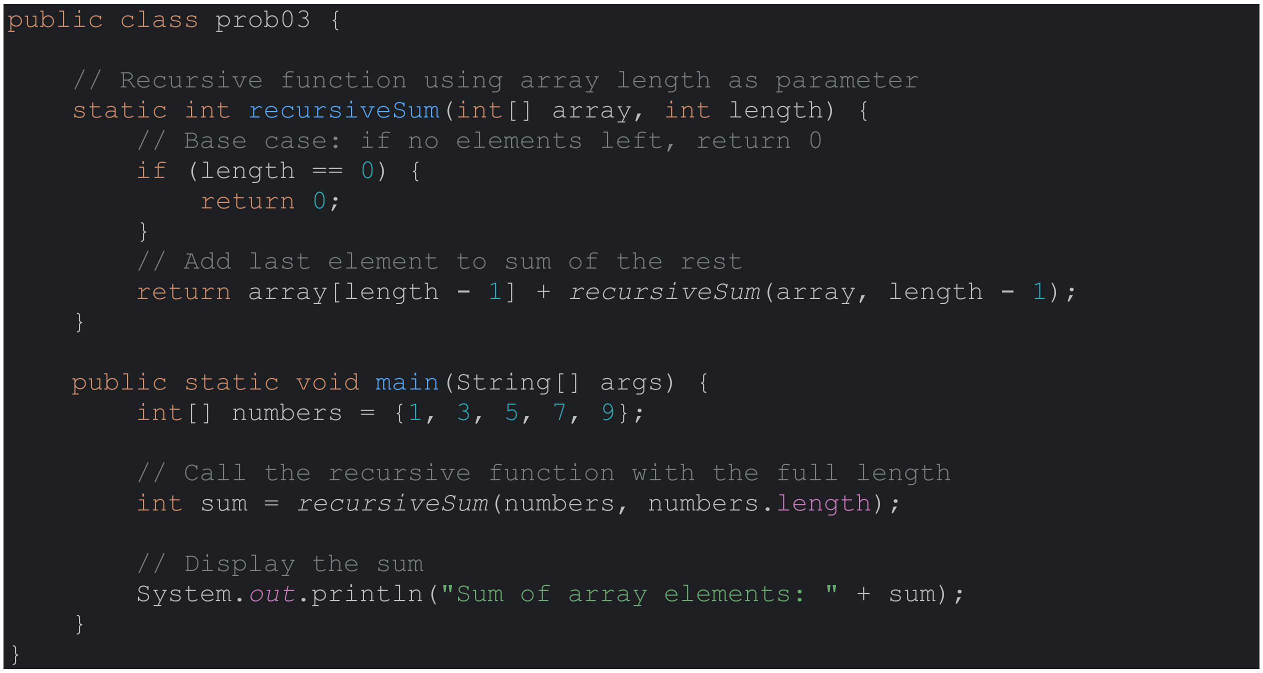
```
public class PROB02 {

    // Custom Stack class using char array
    static class Stack {
        int pointer = -1;   // Top pointer
        int limit;          // Capacity
        char[] store;       // Array to store elements

        // Constructor to initialize stack
        Stack(int size) {
            limit = size;
            store = new char[limit];
        }

        // Push a character to stack
        void push(char c) {
            if (pointer >= limit - 1) {
                System.out.println("Stack Overflow");
                return;
            }
            store[++pointer] = c;
        }

        // Pop a character from stack
        char pop() {
            if (pointer < 0) {
                return '\0';
            }
            return store[pointer--];
        }

        // Check if stack is empty
        boolean isEmpty() {
            return pointer == -1;
        }
    }

    // Function to check if only round parentheses are balanced
    static boolean areParenthesesBalanced(String input) {
        Stack s = new Stack(input.length());

        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);

            // Push if '(' found
            if (c == '(') {
                s.push(c);
            }
            // Pop if ')' found
            else if (c == ')') {
                if (s.isEmpty()) {
                    // Found closing without opening
                    return false;
                }
                s.pop();
            }
            // Ignore all other characters
        }

        // If stack is empty, all '(' matched with ')'
        return s.isEmpty();
    }

    public static void main(String[] args) {
        String expr = "(a+b)+(c*d)-(e/f)";

        if (areParenthesesBalanced(expr)) {
            System.out.println("The parentheses are balanced.");
        } else {
            System.out.println("The parentheses are NOT balanced.");
        }
    }
}
```

OUTPUT;

```
The parentheses are balanced.
```

PROGRAM:3

TO WRITE A PROGRAM TO CALCULATE THE SUM OF ALL INTEGER ELEMENTS IN AN ARRAY BY IMPLEMENTING A RECURSIVE SUM METHOD/FUNCTION



```java
public class prob03 {

    // Recursive function using array length as parameter
    static int recursiveSum(int[] array, int length) {
        // Base case: if no elements left, return 0
        if (length == 0) {
            return 0;
        }
        // Add last element to sum of the rest
        return array[length - 1] + recursiveSum(array, length - 1);
    }

    public static void main(String[] args) {
        int[] numbers = {1, 3, 5, 7, 9};

        // Call the recursive function with the full length
        int sum = recursiveSum(numbers, numbers.length);

        // Display the sum
        System.out.println("Sum of array elements: " + sum);
    }
}
```

OUTPUT;

Sum of array elements: 25