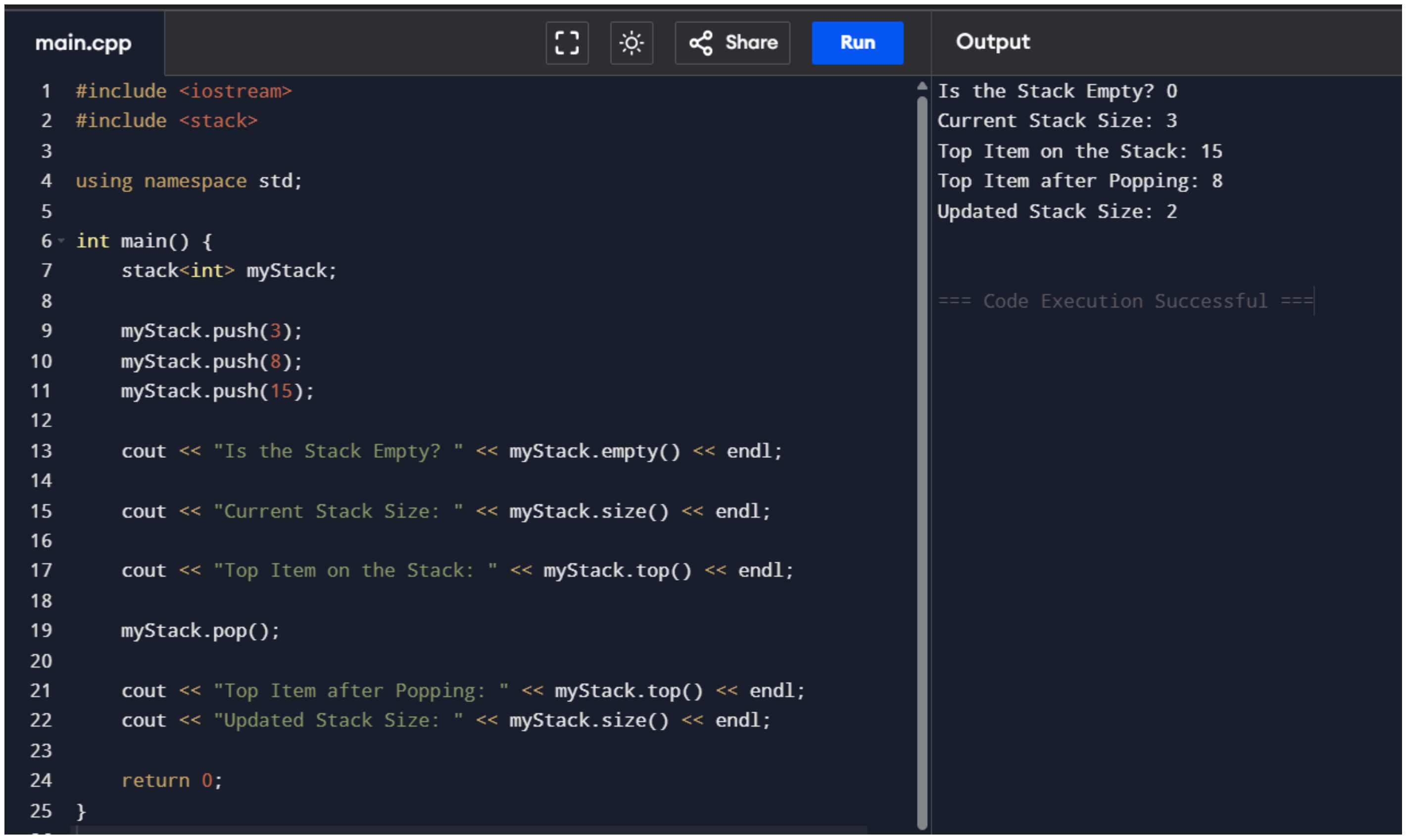
Activity No. 4.1	
Hands-on Activity 4.1 Stacks	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 8/24/2025
Section: CPE21S4	**Date Submitted:** 8/25/2025
Name(s): Avila, Vince Gabriel V.	**Instructor:** Engr. Jimlord Quejado

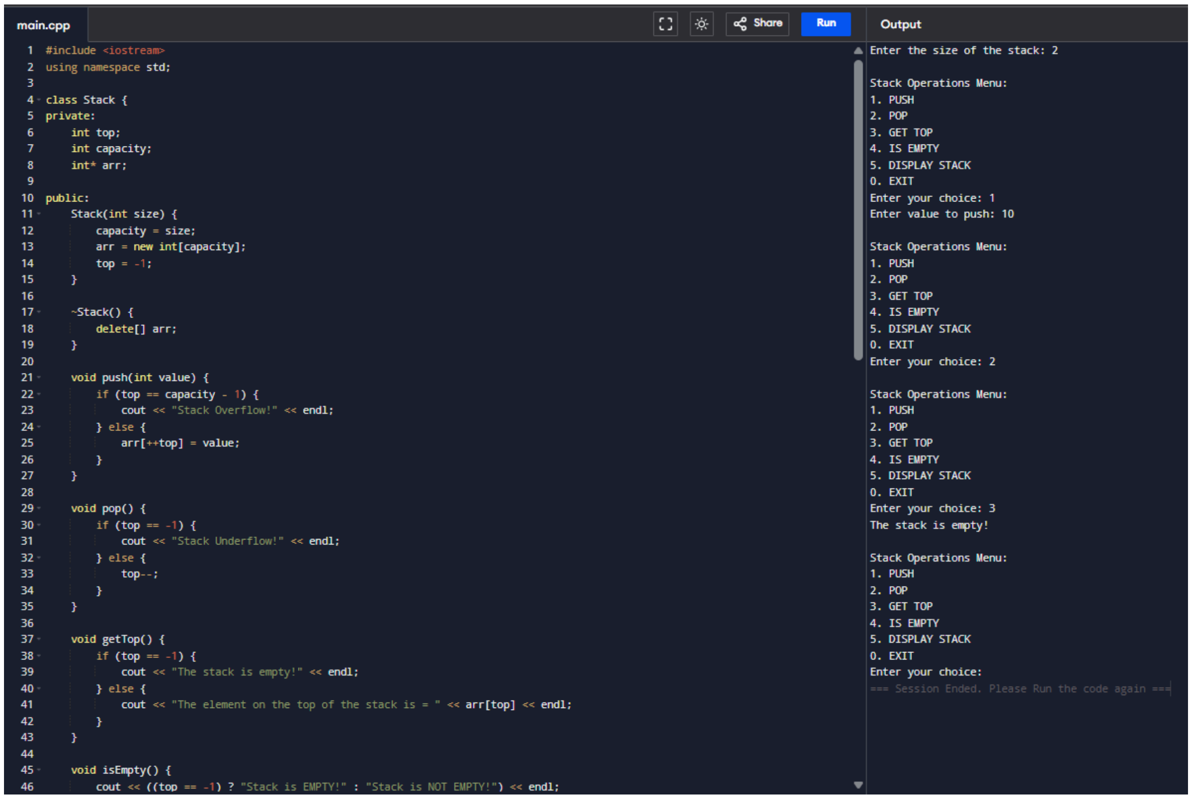
6. Output



```cpp
#include <iostream>
#include <stack>

using namespace std;

int main() {
    stack<int> myStack;

    myStack.push(3);
    myStack.push(8);
    myStack.push(15);

    cout << "Is the Stack Empty? " << myStack.empty() << endl;

    cout << "Current Stack Size: " << myStack.size() << endl;

    cout << "Top Item on the Stack: " << myStack.top() << endl;

    myStack.pop();

    cout << "Top Item after Popping: " << myStack.top() << endl;
    cout << "Updated Stack Size: " << myStack.size() << endl;

    return 0;
}
```

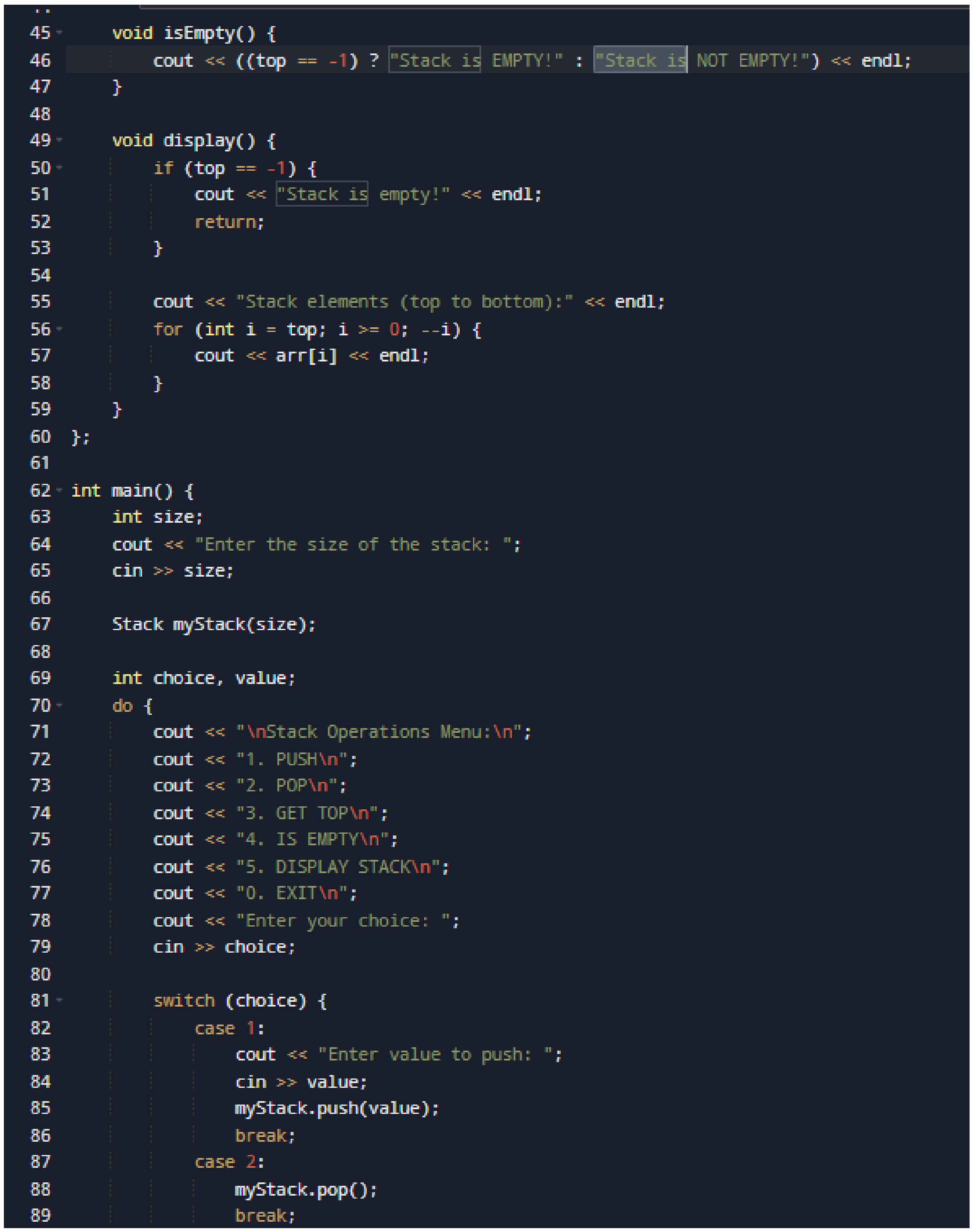
```
Is the Stack Empty? 0
Current Stack Size: 3
Top Item on the Stack: 15
Top Item after Popping: 8
Updated Stack Size: 2


=== Code Execution Successful ===
```

Table 4.1

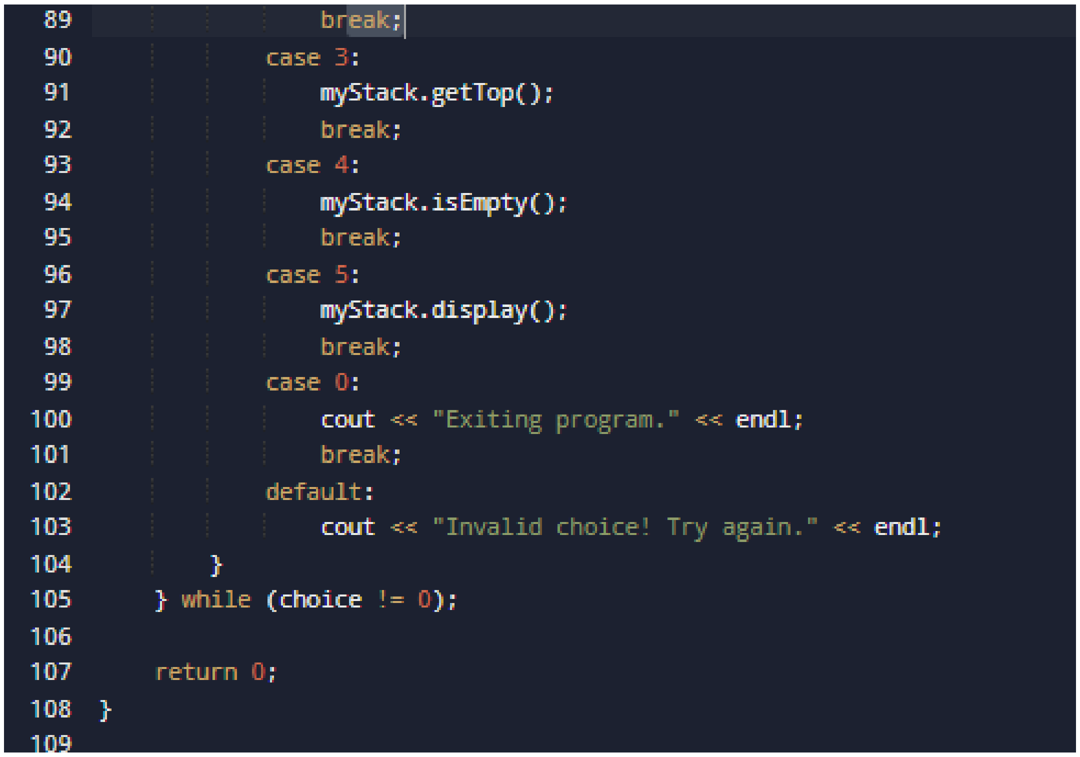


```cpp
#include <iostream>
using namespace std;

class Stack {
private:
    int top;
    int capacity;
    int* arr;

public:
    Stack(int size) {
        capacity = size;
        arr = new int[capacity];
        top = -1;
    }

    ~Stack() {
        delete[] arr;
    }

    void push(int value) {
        if (top == capacity - 1) {
            cout << "Stack Overflow!" << endl;
        } else {
            arr[++top] = value;
        }
    }

    void pop() {
        if (top == -1) {
            cout << "Stack Underflow!" << endl;
        } else {
            top--;
        }
    }

    void getTop() {
        if (top == -1) {
            cout << "The stack is empty!" << endl;
        } else {
            cout << "The element on the top of the stack is = " << arr[top] << endl;
        }
    }

    void isEmpty() {
        cout << ((top == -1) ? "Stack is EMPTY!" : "Stack is NOT EMPTY!") << endl;
```

```
Enter the size of the stack: 2

Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 1
Enter value to push: 10

Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 2

Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice: 3
The stack is empty!

Stack Operations Menu:
1. PUSH
2. POP
3. GET TOP
4. IS EMPTY
5. DISPLAY STACK
0. EXIT
Enter your choice:
=== Session Ended. Please Run the code again ===
```

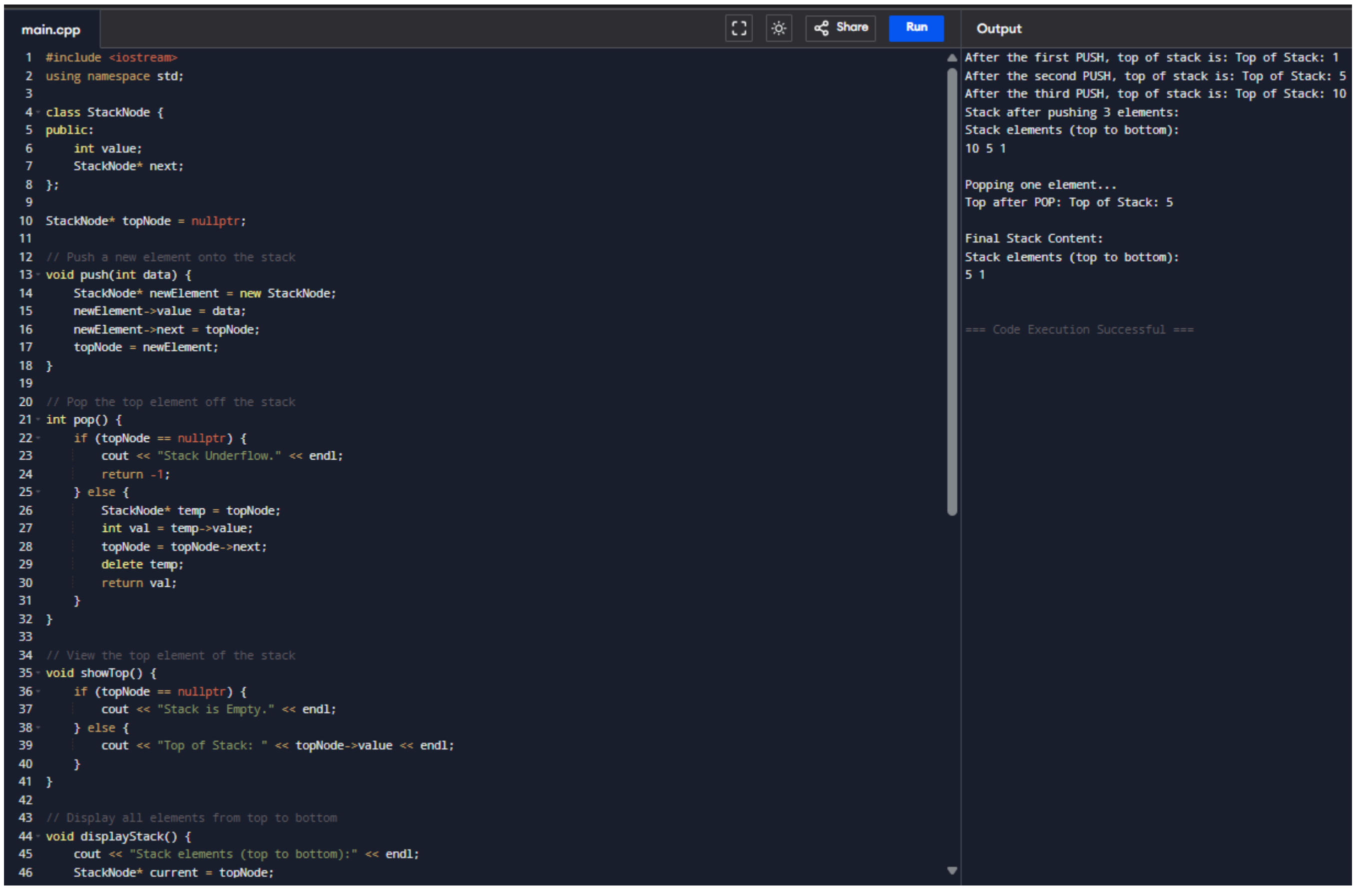


```cpp
    void isEmpty() {
        cout << ((top == -1) ? "Stack is EMPTY!" : "Stack is NOT EMPTY!") << endl;
    }

    void display() {
        if (top == -1) {
            cout << "Stack is empty!" << endl;
            return;
        }

        cout << "Stack elements (top to bottom):" << endl;
        for (int i = top; i >= 0; --i) {
            cout << arr[i] << endl;
        }
    }
};

int main() {
    int size;
    cout << "Enter the size of the stack: ";
    cin >> size;

    Stack myStack(size);

    int choice, value;
    do {
        cout << "\nStack Operations Menu:\n";
        cout << "1. PUSH\n";
        cout << "2. POP\n";
        cout << "3. GET TOP\n";
        cout << "4. IS EMPTY\n";
        cout << "5. DISPLAY STACK\n";
        cout << "0. EXIT\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter value to push: ";
                cin >> value;
                myStack.push(value);
                break;
            case 2:
                myStack.pop();
                break;
```



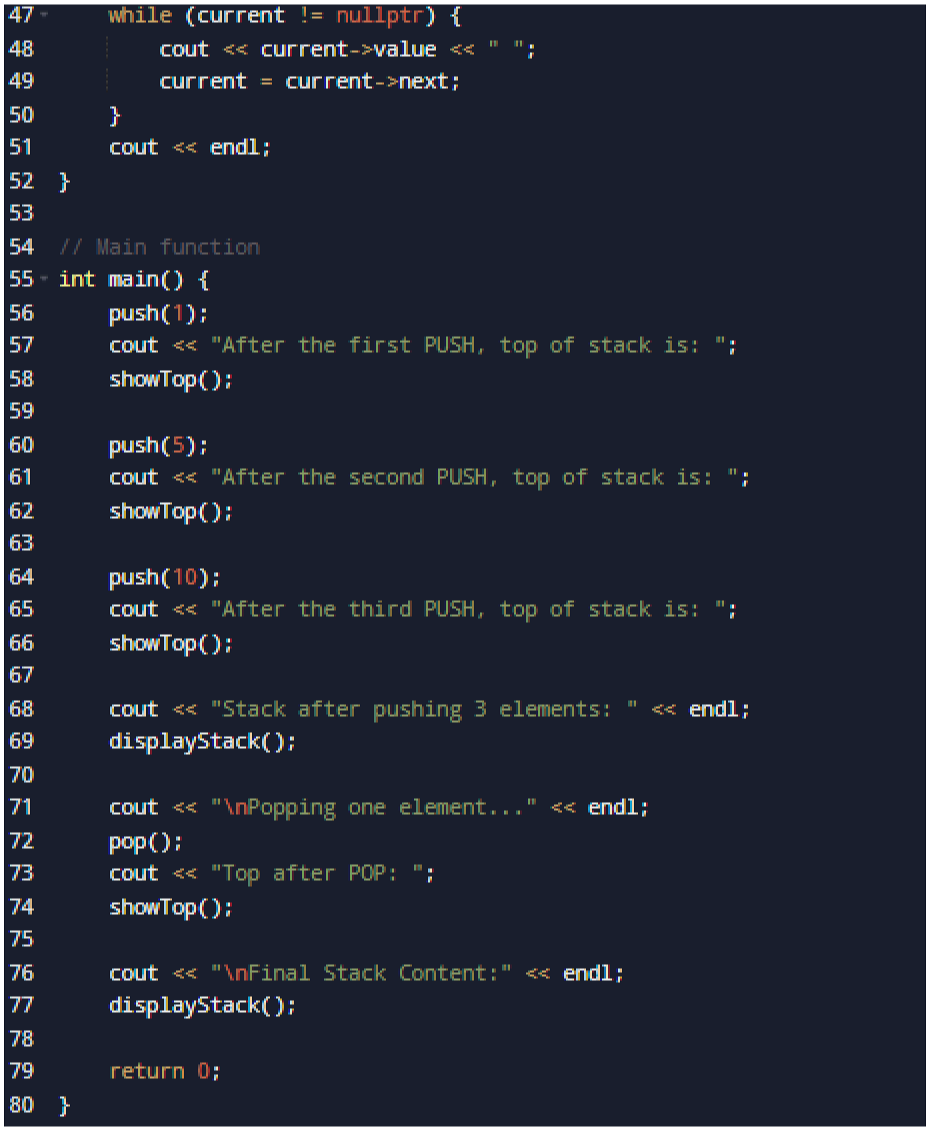
```cpp
                break;
            case 3:
                myStack.getTop();
                break;
            case 4:
                myStack.isEmpty();
                break;
            case 5:
                myStack.display();
                break;
            case 0:
                cout << "Exiting program." << endl;
                break;
            default:
                cout << "Invalid choice! Try again." << endl;
        }
    } while (choice != 0);

    return 0;
}

```

Table 4.2



```cpp
#include <iostream>
using namespace std;

class StackNode {
public:
    int value;
    StackNode* next;
};

StackNode* topNode = nullptr;

// Push a new element onto the stack
void push(int data) {
    StackNode* newElement = new StackNode;
    newElement->value = data;
    newElement->next = topNode;
    topNode = newElement;
}

// Pop the top element off the stack
int pop() {
    if (topNode == nullptr) {
        cout << "Stack Underflow." << endl;
        return -1;
    } else {
        StackNode* temp = topNode;
        int val = temp->value;
        topNode = topNode->next;
        delete temp;
        return val;
    }
}

// View the top element of the stack
void showTop() {
    if (topNode == nullptr) {
        cout << "Stack is Empty." << endl;
    } else {
        cout << "Top of Stack: " << topNode->value << endl;
    }
}

// Display all elements from top to bottom
void displayStack() {
    cout << "Stack elements (top to bottom):" << endl;
    StackNode* current = topNode;
```

```
After the first PUSH, top of stack is: Top of Stack: 1
After the second PUSH, top of stack is: Top of Stack: 5
After the third PUSH, top of stack is: Top of Stack: 10
Stack after pushing 3 elements:
Stack elements (top to bottom):
10 5 1

Popping one element...
Top after POP: Top of Stack: 5

Final Stack Content:
Stack elements (top to bottom):
5 1


=== Code Execution Successful ===
```

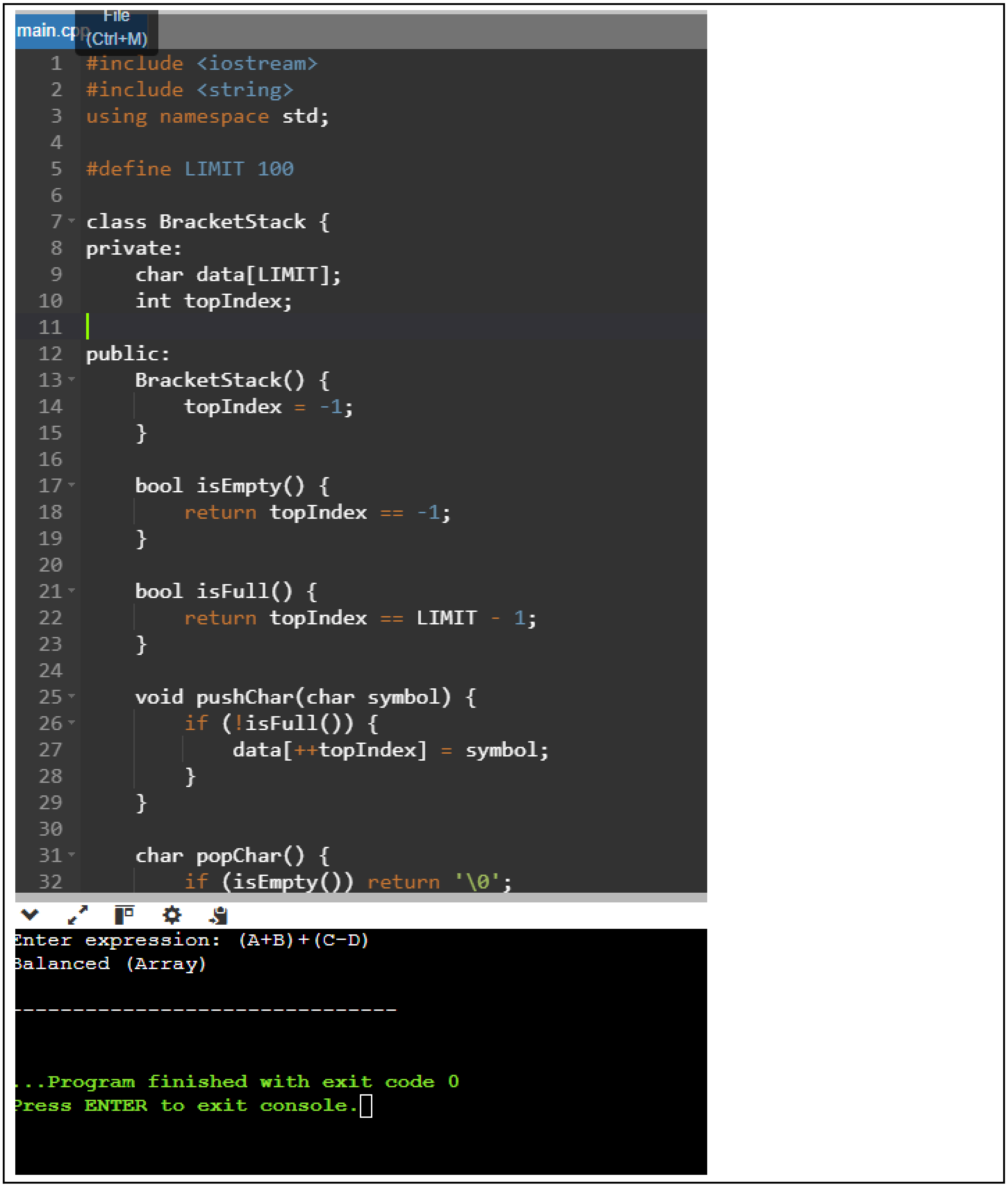


```
    while (current != nullptr) {
        cout << current->value << " ";
        current = current->next;
    }
    cout << endl;
}

// Main function
int main() {
    push(1);
    cout << "After the first PUSH, top of stack is: ";
    showTop();

    push(5);
    cout << "After the second PUSH, top of stack is: ";
    showTop();

    push(10);
    cout << "After the third PUSH, top of stack is: ";
    showTop();

    cout << "Stack after pushing 3 elements: " << endl;
    displayStack();

    cout << "\nPopping one element..." << endl;
    pop();
    cout << "Top after POP: ";
    showTop();

    cout << "\nFinal Stack Content:" << endl;
    displayStack();

    return 0;
}
```

7. Supplementary Activity

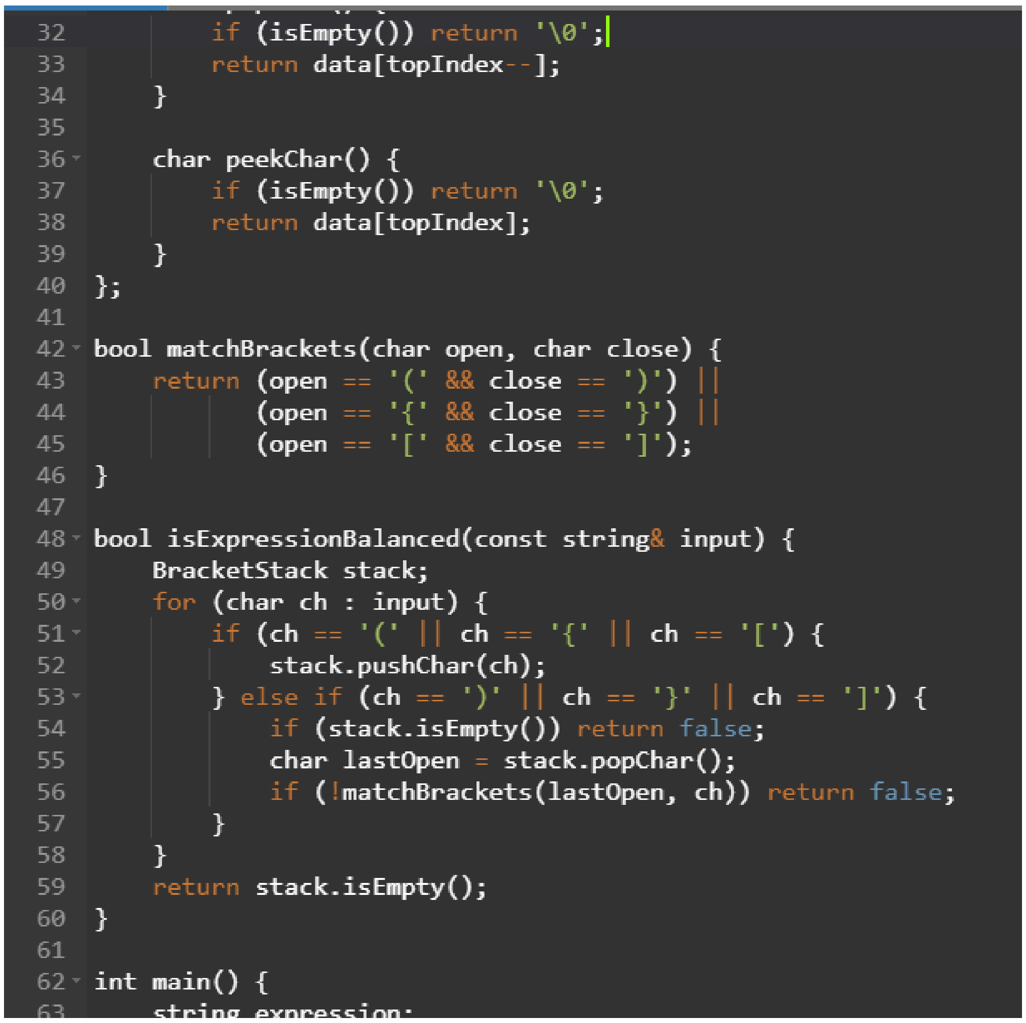
ILO C: SOLVE PROBLEMS USING AN IMPLEMENTATION OF STACK:
Table 4.3

a. **Stack Using Arrays**

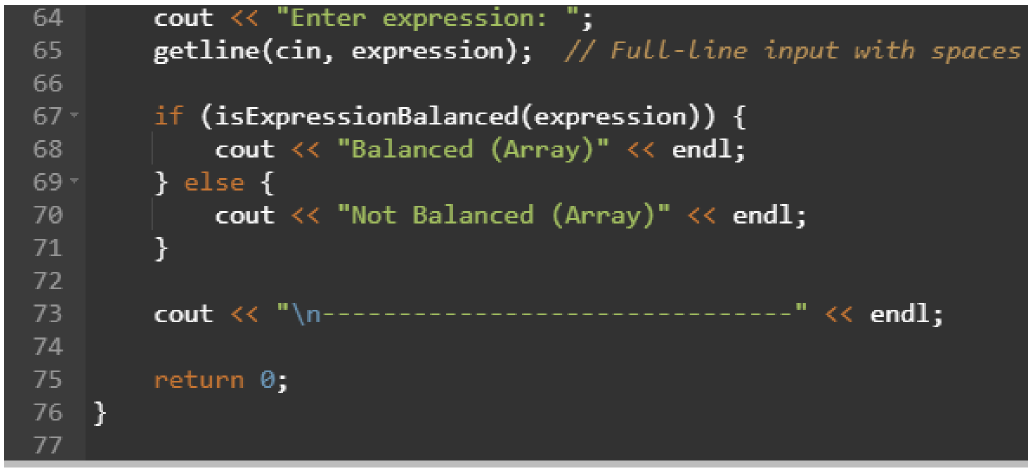


```cpp
#include <iostream>
#include <string>
using namespace std;

#define LIMIT 100

class BracketStack {
private:
    char data[LIMIT];
    int topIndex;

public:
    BracketStack() {
        topIndex = -1;
    }

    bool isEmpty() {
        return topIndex == -1;
    }

    bool isFull() {
        return topIndex == LIMIT - 1;
    }

    void pushChar(char symbol) {
        if (!isFull()) {
            data[++topIndex] = symbol;
        }
    }

    char popChar() {
        if (isEmpty()) return '\0';
```

```
Enter expression: (A+B)+(C-D)
Balanced (Array)

-------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

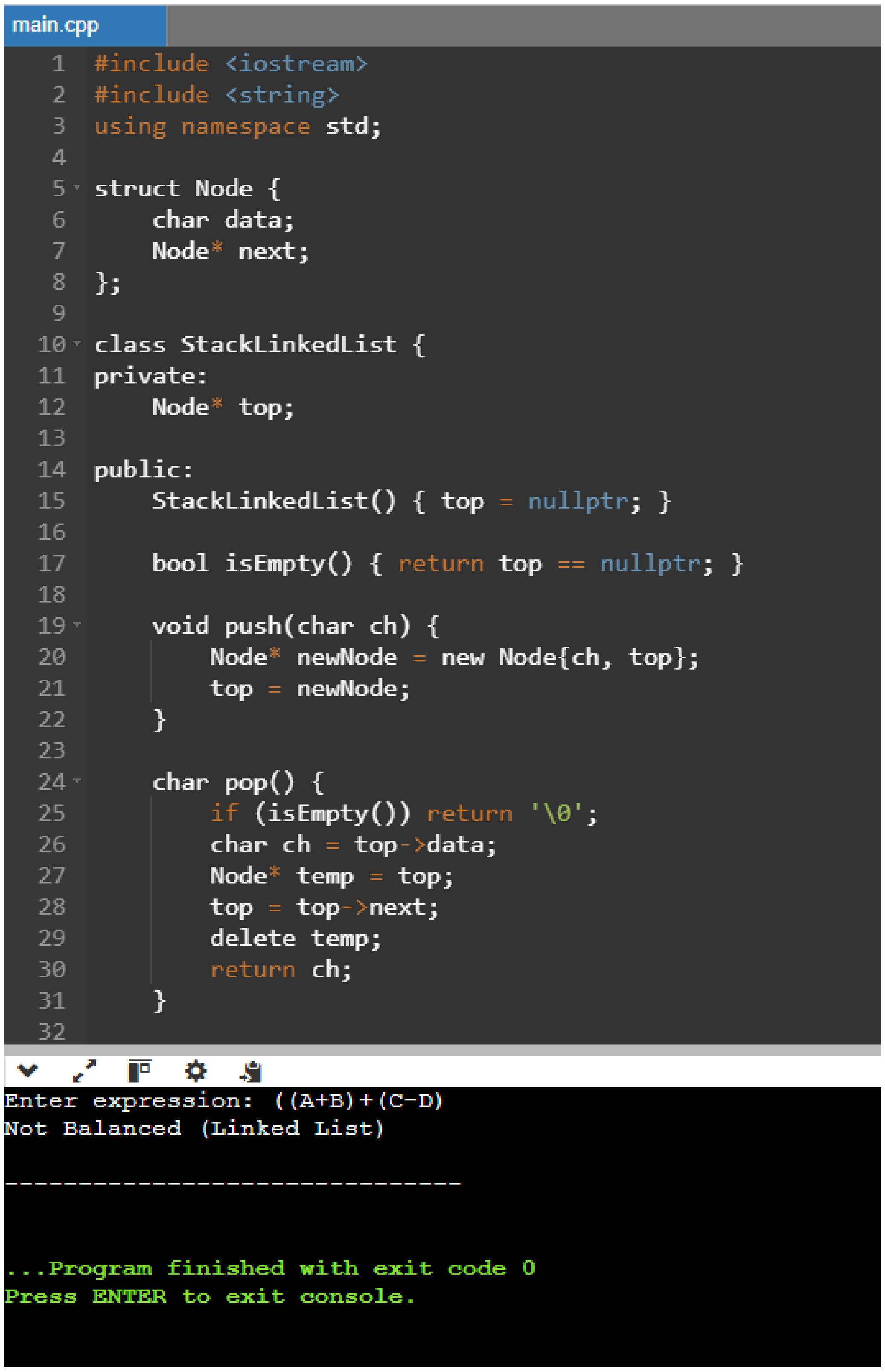


```cpp
        if (isEmpty()) return '\0';
        return data[topIndex--];
    }

    char peekChar() {
        if (isEmpty()) return '\0';
        return data[topIndex];
    }
};

bool matchBrackets(char open, char close) {
    return (open == '(' && close == ')') ||
           (open == '{' && close == '}') ||
           (open == '[' && close == ']');
}

bool isExpressionBalanced(const string& input) {
    BracketStack stack;
    for (char ch : input) {
        if (ch == '(' || ch == '{' || ch == '[') {
            stack.pushChar(ch);
        } else if (ch == ')' || ch == '}' || ch == ']') {
            if (stack.isEmpty()) return false;
            char lastOpen = stack.popChar();
            if (!matchBrackets(lastOpen, ch)) return false;
        }
    }
    return stack.isEmpty();
}

int main() {
    string expression;
```



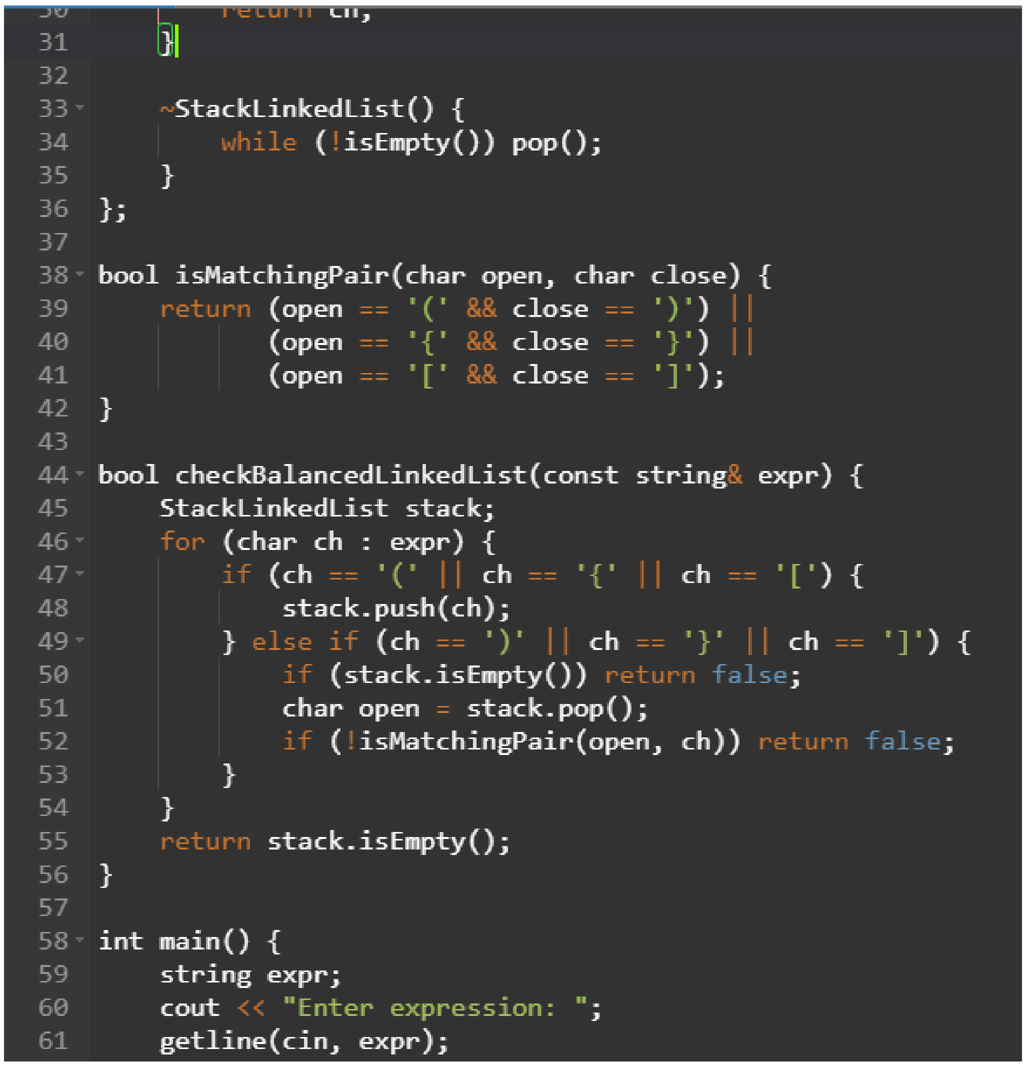
```cpp
    cout << "Enter expression: ";
    getline(cin, expression);  // Full-line input with spaces

    if (isExpressionBalanced(expression)) {
        cout << "Balanced (Array)" << endl;
    } else {
        cout << "Not Balanced (Array)" << endl;
    }

    cout << "\n-------------------------------" << endl;

    return 0;
}
```

b. **Stack Using Linked Lists**

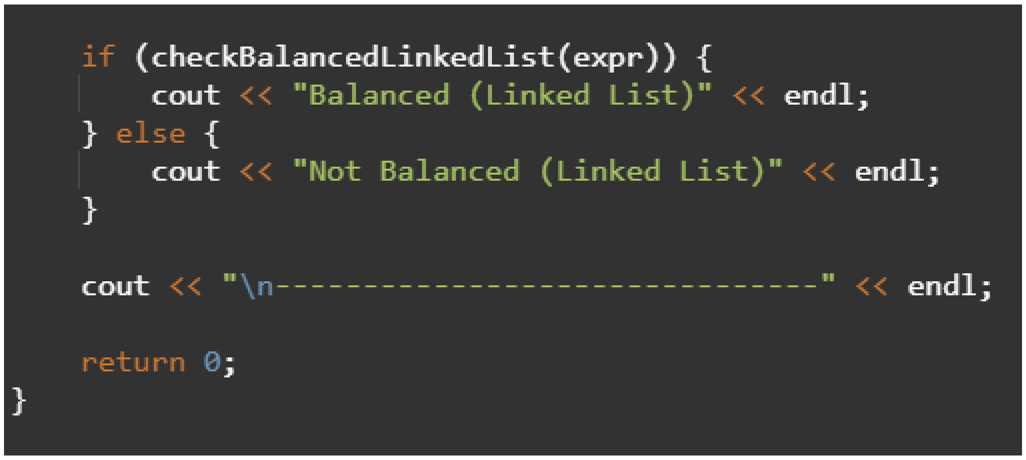

main.cpp

```cpp
#include <iostream>
#include <string>
using namespace std;

struct Node {
    char data;
    Node* next;
};

class StackLinkedList {
private:
    Node* top;

public:
    StackLinkedList() { top = nullptr; }

    bool isEmpty() { return top == nullptr; }

    void push(char ch) {
        Node* newNode = new Node{ch, top};
        top = newNode;
    }

    char pop() {
        if (isEmpty()) return '\0';
        char ch = top->data;
        Node* temp = top;
        top = top->next;
        delete temp;
        return ch;
    }

```

```
Enter expression: ((A+B)+(C-D)
Not Balanced (Linked List)

-------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

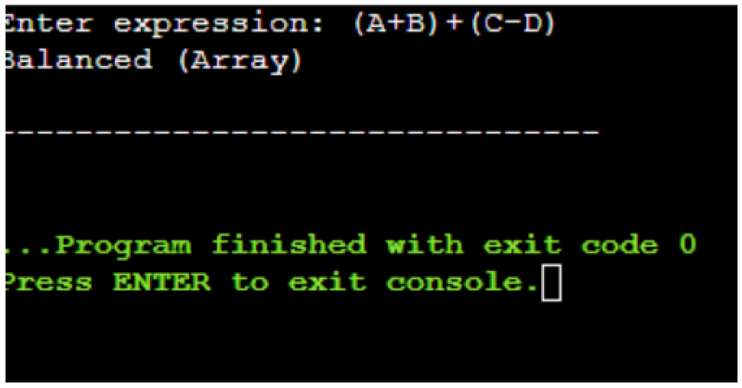


```cpp
        return ch;
    }

    ~StackLinkedList() {
        while (!isEmpty()) pop();
    }
};

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

bool checkBalancedLinkedList(const string& expr) {
    StackLinkedList stack;
    for (char ch : expr) {
        if (ch == '(' || ch == '{' || ch == '[') {
            stack.push(ch);
        } else if (ch == ')' || ch == '}' || ch == ']') {
            if (stack.isEmpty()) return false;
            char open = stack.pop();
            if (!isMatchingPair(open, ch)) return false;
        }
    }
    return stack.isEmpty();
}

int main() {
    string expr;
    cout << "Enter expression: ";
    getline(cin, expr);
```



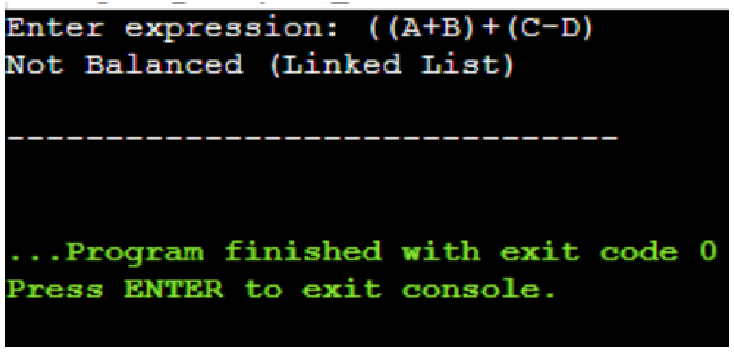
```cpp
    if (checkBalancedLinkedList(expr)) {
        cout << "Balanced (Linked List)" << endl;
    } else {
        cout << "Not Balanced (Linked List)" << endl;
    }

    cout << "\n-------------------------------" << endl;

    return 0;
}
```

Self-Checking: Expression: (A+B)+(C-D)



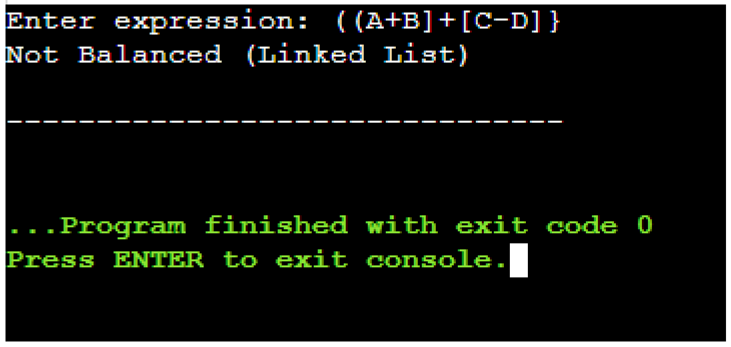
```
Enter expression:  (A+B)+(C-D)
Balanced (Array)

--------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

Self-Checking: Expression: ((A+B)+(C-D)



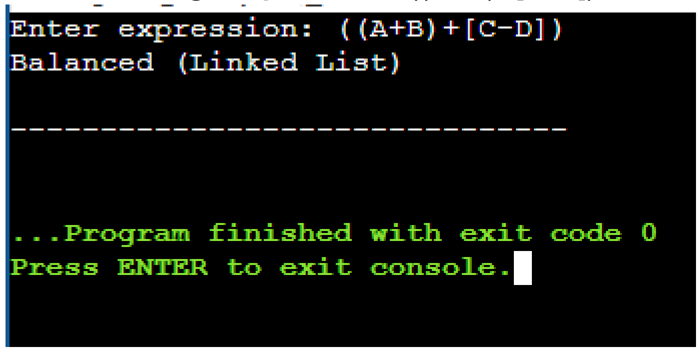
```
Enter expression:  ((A+B)+(C-D)
Not Balanced (Linked List)

--------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

Self-Checking: Expression: ((A+B)+(C-D)



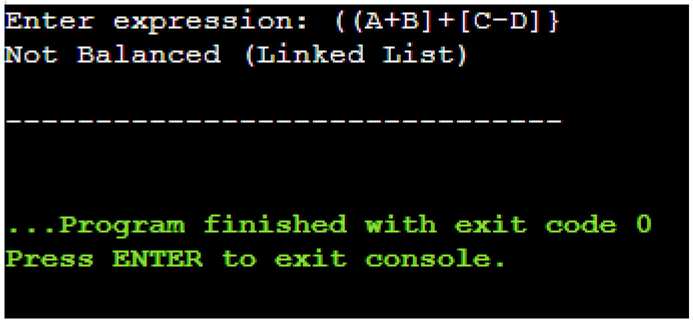
```
Enter expression:  ((A+B]+[C-D]}
Not Balanced (Linked List)

--------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

Self-Checking: Expression: ((A+B)+[C-D])



```
Enter expression:  ((A+B)+[C-D])
Balanced (Linked List)

--------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

Self-Checking: Expression: ((A+B]+[C-D]}



```
Enter expression: ((A+B]+[C-D]}
Not Balanced (Linked List)

-------------------------------


...Program finished with exit code 0
Press ENTER to exit console.
```

8. Conclusion

In conclusion, I am still learning about how a stack works by following the Last In, First Out rule and how the different operations push, pop, top, isEmpty, and display are used. I am beginning to understand how data is stored and removed in the order it was put in. I still have to work on organizing and writing better code. The activities did help me better understand how stacks work in a program.

9. Assessment Rubric