Activity No. <4.1>	
<Hands-on Activity 4.1 Stacks>	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 8/25/25**
Section: CPE 010-CPE21S4	**Date Submitted: 8/25/25**
Name(s): Catungal Kerwin Jan B	**Instructor: Jimlord Quejado**

6. Output

ILO A: Create a stack using the C++ STL

*] ILOA1.cpp

```cpp
#include <iostream>
#include <stack> // Calling Stack from the STL

using namespace std;

int main()
{
    stack<int> newStack;

    newStack.push(3); //Adds 3 to the stack
    newStack.push(8);
    newStack.push(15);

    // returns a boolean response depending on if the stack is empty or not
    cout << "Stack Empty? " << newStack.empty() << endl;

    // returns the size of the stack itself
    cout << "Stack Size: " << newStack.size() << endl;

    // returns the topmost element of the stack
    cout << "Top Element of the Stack: " << newStack.top() << endl;

    // removes the topmost element of the stack
    newStack.pop();

    cout << "Top Element of the Stack: " << newStack.top() << endl;
    cout << "Stack Size: " << newStack.size() << endl;

    return 0;
}
```

C:\Users\Olaco\Downloads\ILOA1.exe

```
Stack Empty? 0
Stack Size: 3
Top Element of the Stack: 15
Top Element of the Stack: 8
Stack Size: 2

--------------------------------
Process exited after 0.3566 seconds with return value 0
Press any key to continue . . .
```

Table 4-2. Output of ILO B.1.

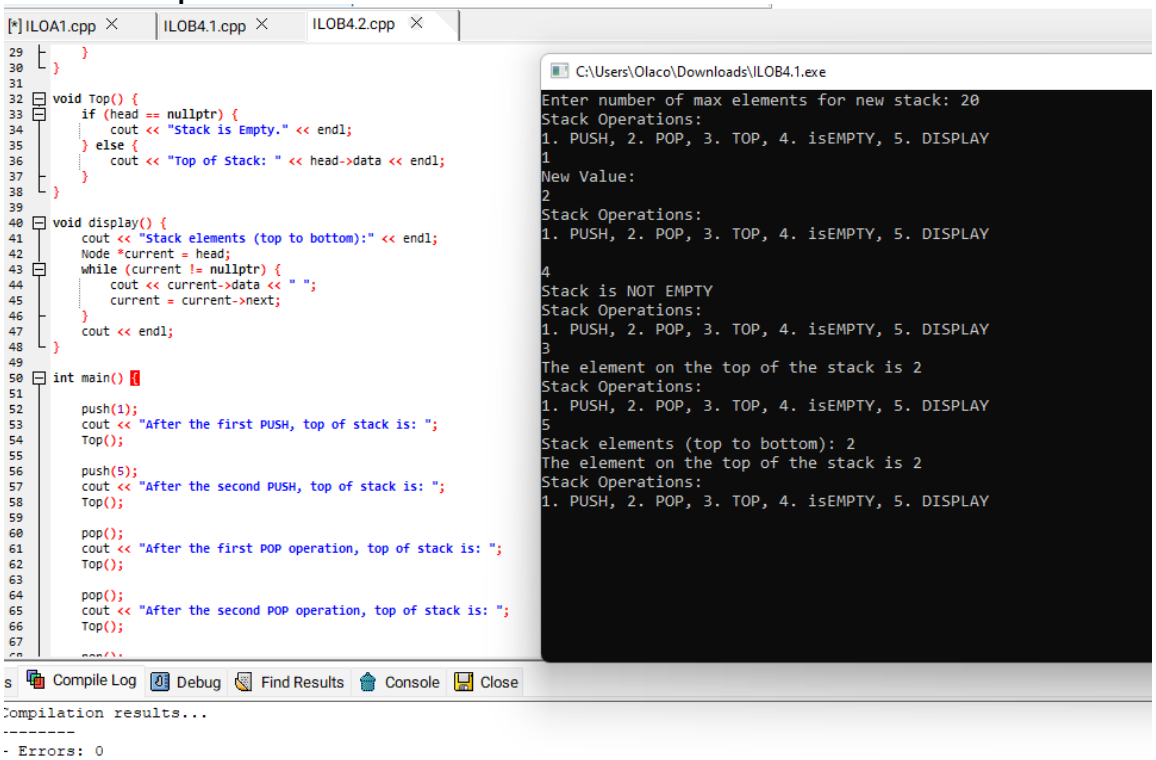
LOA1.cpp | ILOB4.1.cpp

```cpp
    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;
    }

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

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

int main() {
    int size;
    cout << "Enter number of max elements for new stack: ";
    cin >> size;

    Stack s(size);
    int choice, value;

    while (true) {
        cout << "Stack Operations:\n";
        cout << "1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY\n";
```

C:\Users\Olaco\Downloads\ILOB4.1.exe

```
Enter number of max elements for new stack: 20
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
1
New Value:
2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
4
Stack is NOT EMPTY
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
3
The element on the top of the stack is 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
5
Stack elements (top to bottom): 2
The element on the top of the stack is 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
```

Table 4-3. Output of ILO B.2.

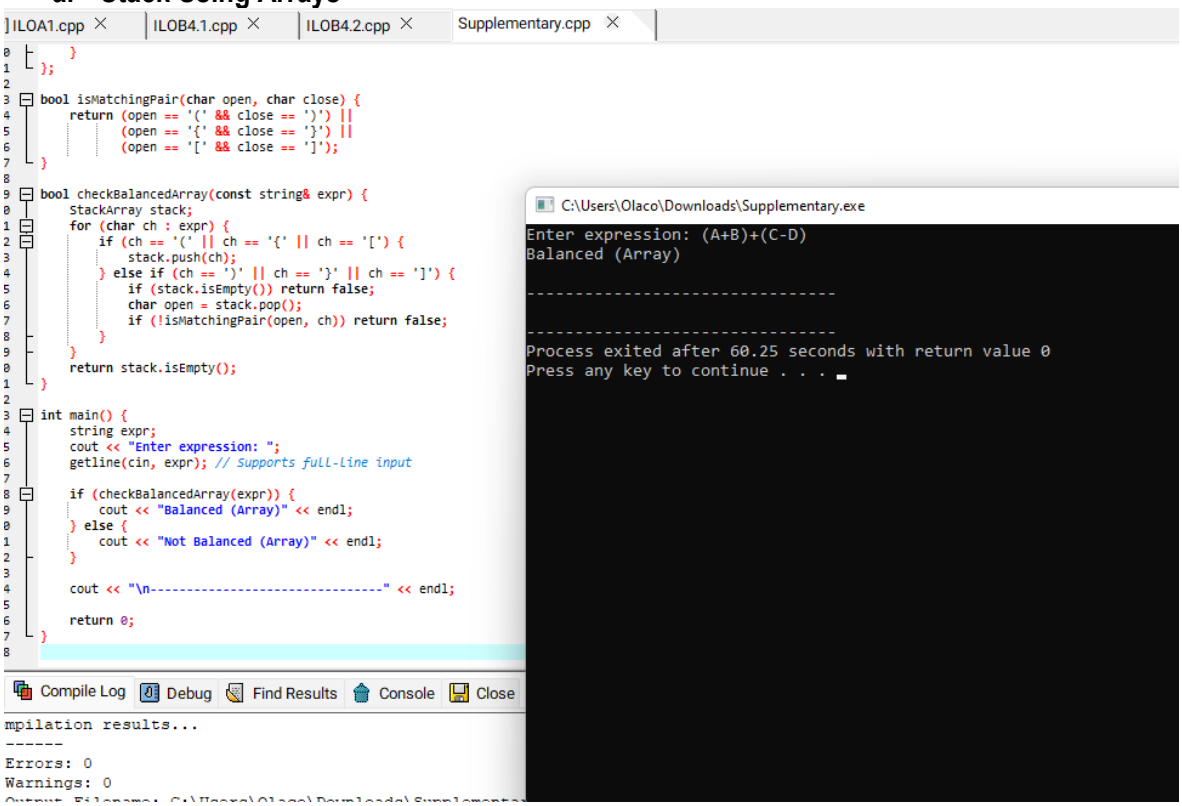


7. Supplementary Activity

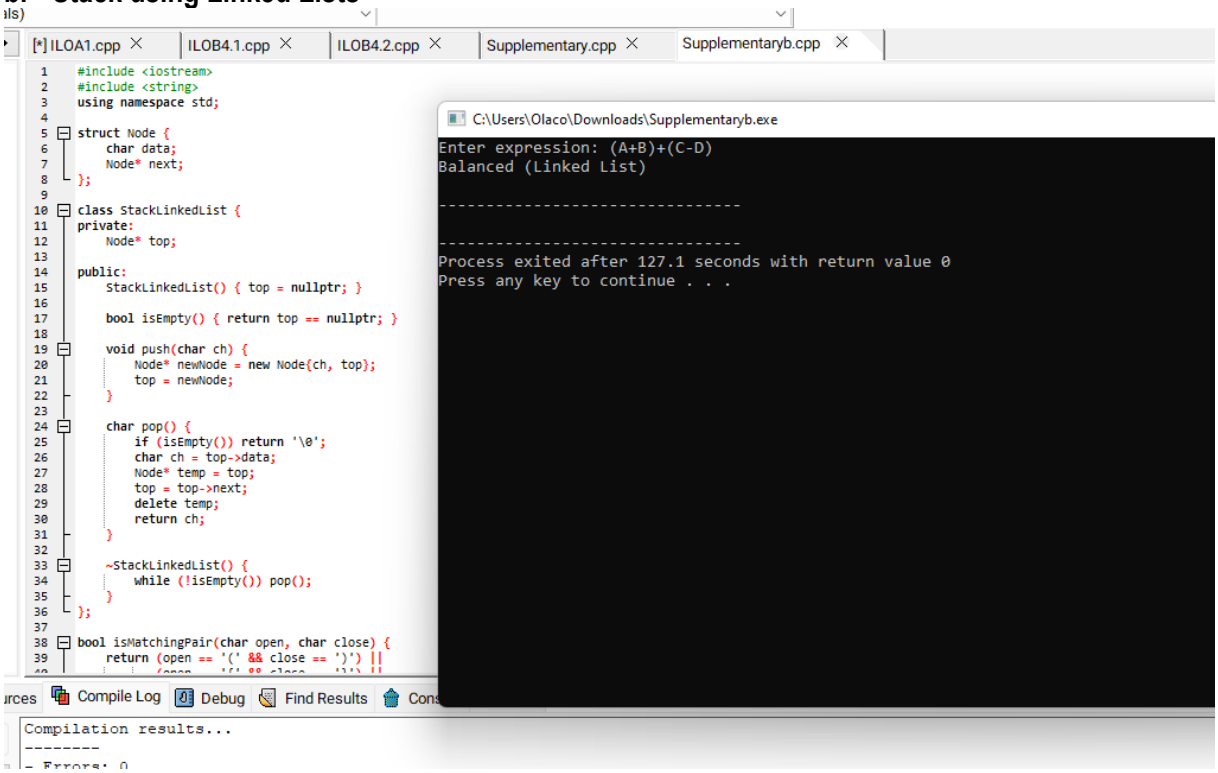
ILO C: SOLVE PROBLEMS USING AN IMPLEMENTATION OF STACK:

Table 4.3

a. Stack Using Arrays



b. Stack using Linked Lists



```
[*] ILOA1.cpp | ILOB4.1.cpp | ILOB4.2.cpp | Supplementary.cpp | Supplementaryb.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;
    }

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

bool isMatchingPair(char open, char close) {
    return (open == '(' && close == ')') ||

C:\Users\Olaco\Downloads\Supplementaryb.exe
Enter expression: (A+B)+(C-D)
Balanced (Linked List)

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

--------------------------------
Process exited after 127.1 seconds with return value 0
Press any key to continue . . .

Compile Log | Debug | Find Results

Compilation results...
--------
```

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

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

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

--------------------------------
Process exited after 27.77 seconds with return value 0
Press any key to continue . . .
```

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

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

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

--------------------------------
Process exited after 29.23 seconds with return value 0
Press any key to continue . . .
```

((A+B)+[C-D])

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

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

--------------------------------
Process exited after 4.137 seconds with return value 0
Press any key to continue . . .
```

((A+B)+[C-D])

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

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

--------------------------------
Process exited after 3.992 seconds with return value 0
Press any key to continue . . .
```

8. Conclusion

In this activity we used push, pop, top, is empty, display. this process helped me learn how data is stored and viewed. It follows the rules of the Last In, First Out rule or LIFO. It took some logical thinking to make sure that each one was done correctly and fast. In conclusion, I learned a lot about the main ideas of stacks, but I still need to get better at putting my code in order.

9. Assessment Rubric