Activity No. 4	
Stacks	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed: 04 - 10 - 2024**
Section: CpE21 S4	**Date Submitted: 04 - 10 - 2024**
Name(s):Bonifacio, Nyko Adrein	**Instructor:Prof. Sayo**

6. Output

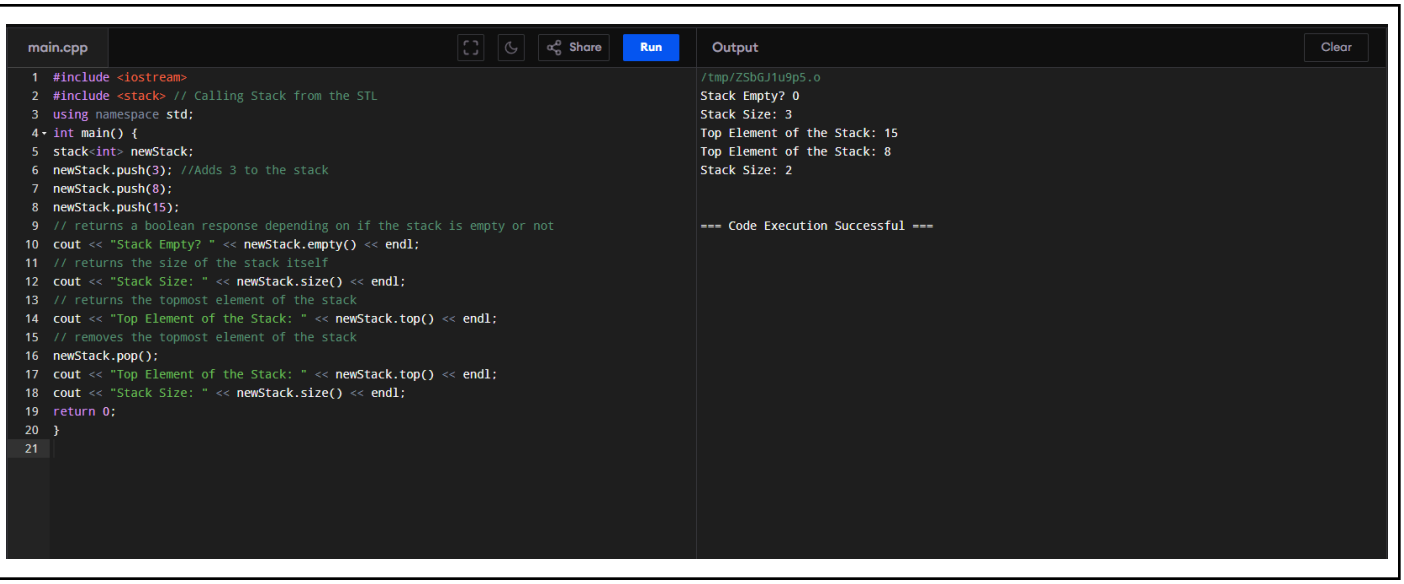


Table 4-1. Output of ILO A

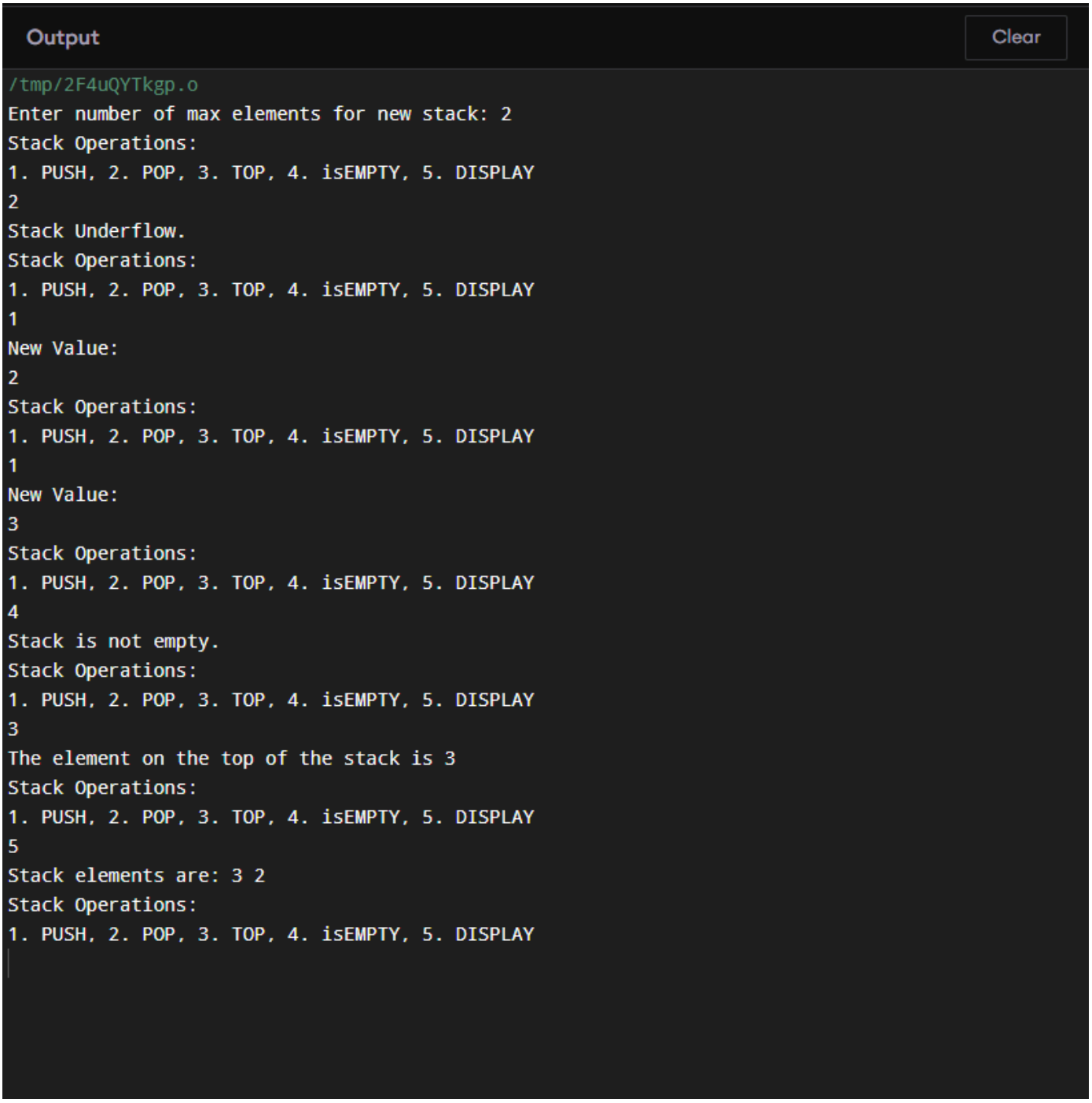


Table 4-2. Output of ILO B.1

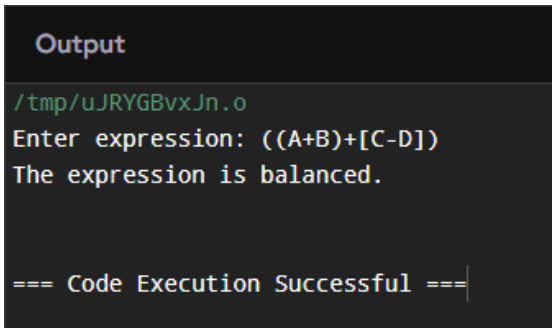
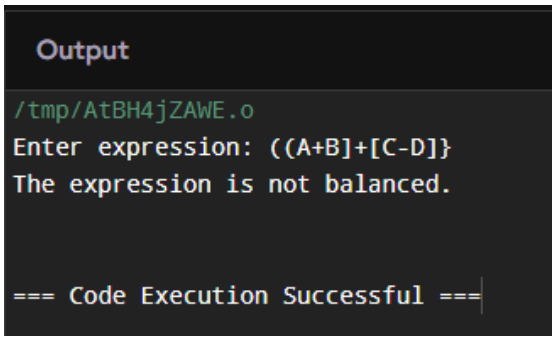
```
Output                                                    Clear
/tmp/fYor08m5L7.o
After the first PUSH, top of stack is: Top of Stack: 1
Stack elements: 1
After the second PUSH, top of stack is: Top of Stack: 5
Stack elements: 5 1
After the first POP operation, top of stack is: Top of Stack: 1
Stack elements: 1
After the second POP operation, top of stack is: Stack is Empty.
Stack is Empty.
Stack Underflow.


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

Table 4-3. Output of ILO B.2

7. Supplementary Activity

EXPRESSION	VALID (Y/N?)	OUTPUT (CONSOLE SCREEN SHOT)	ANALYSIS
(A+B)+(C-D)	Y	Output /tmp/0p949UyHqG.o Enter expression: (A+B)+(C-D) The expression is balanced. === Code Execution Successful ===	Parentheses are properly balanced and in the correct order.
((A+B)+(C-D)	N	Output /tmp/bdWxDHQynK.o Enter expression: ((A+B)+(C-D) The expression is not balanced. === Code Execution Successful ===	Nested parentheses are incorrectly matched and unbalanced.

((A+B)+[C-D])	Y		Parentheses and square brackets are correctly matched and balanced.
((A+B]+[C-D]}	N		Mismatched parentheses and brackets make the expression invalid.

8. Conclusion

In this activity, I learned how to implement stacks using arrays, linked lists, and the C++ STL to check for balanced symbols in expressions. The array-based stack is simple but limited by fixed memory, while the linked list provides dynamic allocation, though it's more complex to manage. The STL stack simplifies the implementation by handling memory automatically. Testing different expressions helped solidify my understanding of handling nested symbols and edge cases. I performed well in applying these concepts, but I can improve by optimizing code, handling more complex edge cases, and providing better error messages.

9. Assessment Rubric