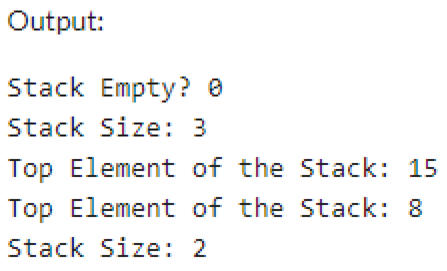
Hands-on Activity 4.1	
Stacks	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** Oct 4, 2024
Section: CPE21S4	**Date Submitted:** Oct 5, 2024
Name(s): LEE IVAN TITONG	**Instructor: Ma'am Maria Rizette Sayo**

6. Output

The code demonstrates basic stack operations using C++'s STL. It creates a stack of integers, pushes values onto it, checks if it's empty, retrieves its size, and accesses the top element. The pop method removes the top item, updating both the stack's size and top element. The stack follows a Last In, First Out (LIFO) structure, making all operations efficient with constant time complexity.

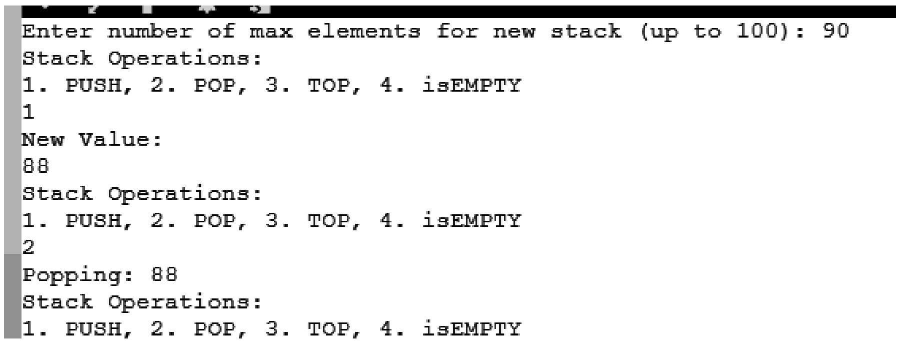


```
Output:

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

Table 4-1. Output of ILO A

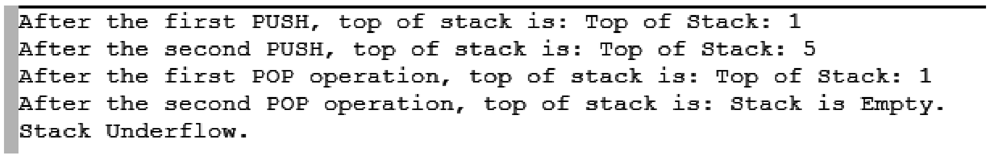
The stack operations include push, which adds a new element to the top if there's space, and pop, which removes the top element if the stack isn't empty. The Top function shows the current top element without removing it, while isEmpty checks if the stack is empty. The new display function prints all elements in the stack from bottom to top, or notifies if the stack is empty.



```
Enter number of max elements for new stack (up to 100): 90
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
1
New Value:
88
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
2
Popping: 88
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
```

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

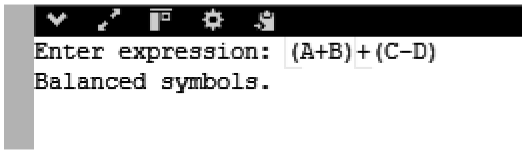
The stack operations include push, which adds a new element to the top of the stack by creating a new node and adjusting the head pointer; pop, which removes and returns the top element, checking for underflow if the stack is empty; Top, which displays the current top element without removing it; and display, which prints all elements in the stack from top to bottom, indicating if the stack is empty. Together, these operations maintain the stack's Last In, First Out (LIFO) structure.



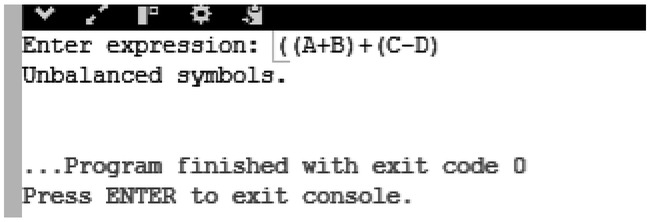
```
After the first PUSH, top of stack is: Top of Stack: 1
After the second PUSH, top of stack is: Top of Stack: 5
After the first POP operation, top of stack is: Top of Stack: 1
After the second POP operation, top of stack is: Stack is Empty.
Stack Underflow.
```

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

7. Supplementary Activity

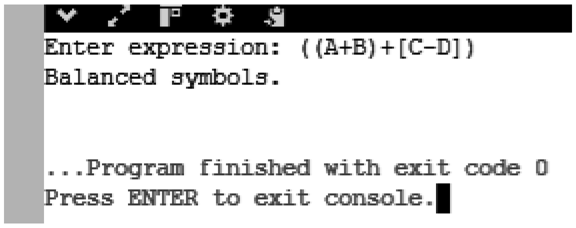


```
Enter expression: (A+B)+(C-D)
Balanced symbols.
```



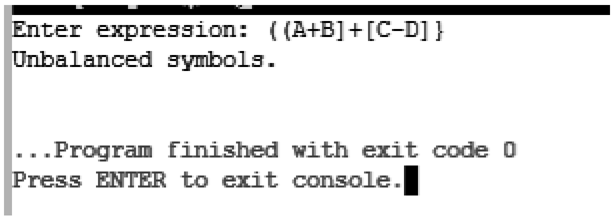
```
Enter expression: ((A+B)+(C-D)
Unbalanced symbols.


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



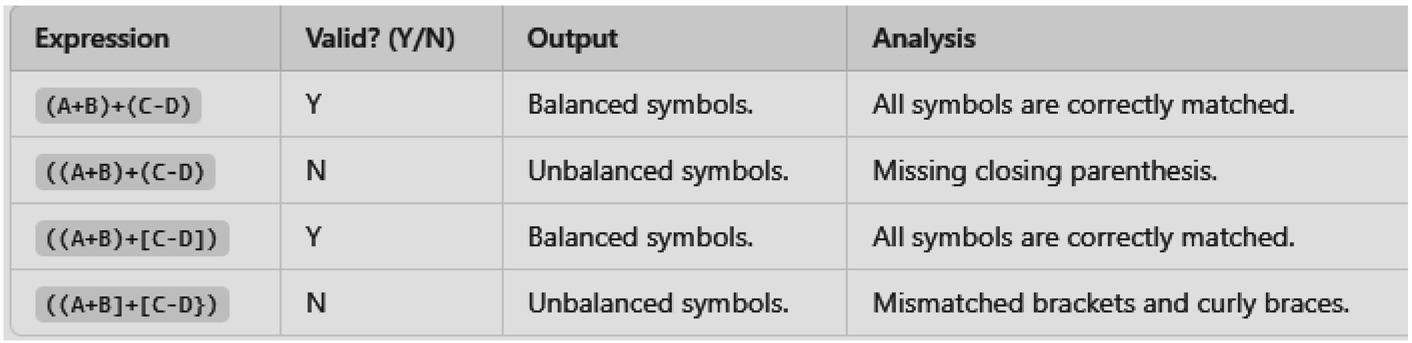
```
Enter expression: ((A+B)+[C-D])
Balanced symbols.


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



```
Enter expression: ((A+B]+[C-D]}
Unbalanced symbols.


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



Expression	Valid? (Y/N)	Output	Analysis
`(A+B)+(C-D)`	Y	Balanced symbols.	All symbols are correctly matched.
`((A+B)+(C-D)`	N	Unbalanced symbols.	Missing closing parenthesis.
`((A+B)+[C-D])`	Y	Balanced symbols.	All symbols are correctly matched.
`((A+B]+[C-D})`	N	Unbalanced symbols.	Mismatched brackets and curly braces.

8. Conclusion

In conclusion, the stack implementations using both an array and a linked list effectively demonstrate key stack operations: `push`, `pop`, `Top`, `isEmpty`, and `display`. These operations manage the stack's contents while following the Last In, First Out (LIFO) principle. The array-based stack is efficient for fixed sizes, while the linked-list version allows for dynamic growth. Together, these examples highlight the usefulness of stacks in organizing and managing data in various applications.

9. Assessment Rubric