Activity No. 4	
Hands-on Activity 4.1 Stacks	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** 10/04/2024
Section: CPE21S4	**Date Submitted:** 10/06/2024
Name(s): Edson Ray E. San Juan	**Instructor:** Prof. Maria Rizette Sayo

6. Output

Output:

```
/tmp/pj1wkNaxSp.o
Stack Empty? 0
Stack Size: 3
Top Element of the Stack: 15
Top Element of the Stack: 8
Stack Size: 2
```

Observation:

Based on my observation the program uses the C++ STL stack library to test basic stack operations. It pushes three elements, checks if the stack is empty, prints the stack size, shows the top element, and then pops the top element. The stack behaves correctly, showing size and top element updates after each operation.

Table 4-1. Output of ILO A

Output:

```
/tmp/FIFG7rGXfb.o
Enter number of max elements for new stack: 34
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
2
Stack Underflow.
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
3
Stack is Empty.
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
1
New Value:
23
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
3
The element on the top of the stack is 23
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
4
0
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
```

Observation:

Based on my observation on B.1. Stacks using Arrays code, the program seems to run in an infinite loop without an exit option. The isEmpty, push, pop, and Top functions work correctly, with appropriate checks for underflow and overflow. This code can be more improve my utilizing the user-defined capacity dynamically, implementation of a break to exit the loop, and add error handling for invalid user input

Table 4-2. Output of ILO B.1

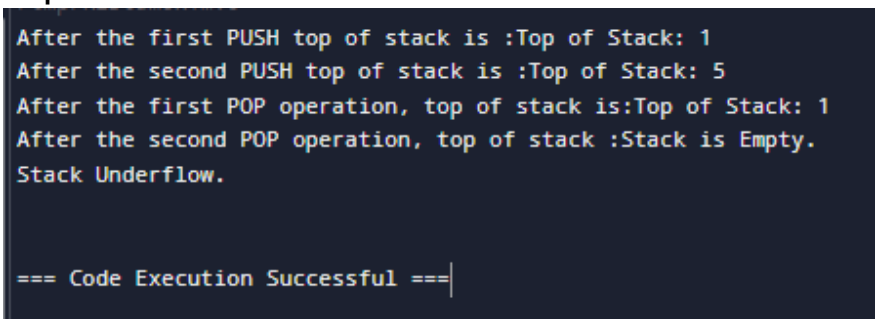
Output:	**Observation:**
	Based on my observation the code implements a stack using a linked list, but the output isn't fully correct. There's redundant code in push, and in pop, resetting both head and tail causes issues. After popping, the top of the stack isn't updated properly, leading to incorrect output. While the "Stack Underflow" message works, the stack behavior needs fixing for accurate results.

Table 4-3. Output of ILO B.2

7. Supplementary Activity

ILO C: Solve problems using an implementation of stack:
The following problem definition and algorithm is provided for checking balancing of symbols. Create an implementation using stacks. Your output must include the following:

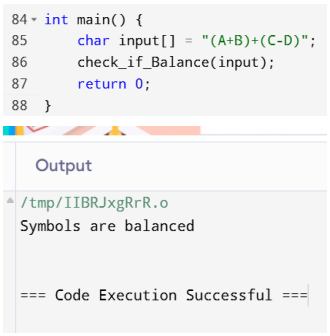
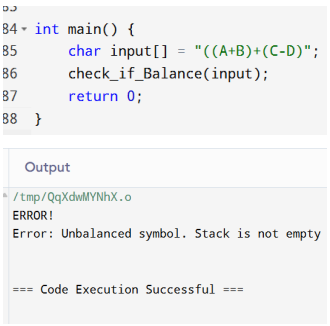
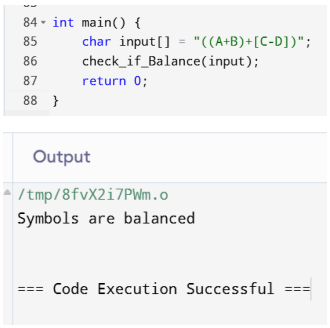
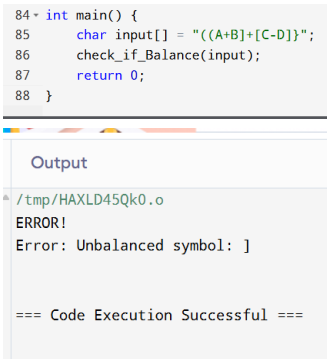
a. Stack using Arrays
b. Stack using Linked Lists
c. (Optional) Stack using C++ STL

Problem Definition:
Stacks can be used to check whether the given expression has balanced symbols. This algorithm is very useful in compilers. Each time the parser reads one character at a time. If the character is an opening delimiter such as (, {, or [- then it is written to the stack. When a closing delimiter is encountered like), }, or]-the stack is popped. The opening and closing delimiters are then compared. If they match, the parsing of the string continues. If they do not match, the parser indicates that there is an error on the line.

Steps:
1. Create a Stack.
2. While(end of input is not reached) {
   a) If the character read is not a symbol to be balanced, ignore it.
   b) If the character is an opening symbol, push it onto the stack.
   c) If it is a closing symbol:
      i) Report an error if the stack is empty.
      ii) Otherwise, pop the stack.
   d) If the symbol popped is not the corresponding opening symbol, report an error.
3. At the end of input, if the stack is not empty: report an error.

Self-Checking:
For the following cases, complete the table using the code you created.

Expression	**Valid? (Y/N)**	**Output (Console Screenshot)**	**Analysis**

(A+B)+(C-D)	Y		Based on my analysis it is a balanced expression because it has a matching pair of opening and closing brackets for each type of bracket, and 'A+B' and 'C-D' is a valid expression.
((A+B)+(C-D)	N		Based on my analysis this is not a balanced expression because the opening bracket '(' that has not been closed. Which results that the stack will not be empty at the end of the string, indicating that the brackets are not properly balanced.
((A+B)+[C-D])	Y		Based on my analysis this is a balanced expression Because the stack is empty after processing the entire string, the brackets and parentheses are properly balanced.
((A+B]+[C-D]}	N		Based on my analysis, this is not a balanced expression, due to mismatched brackets, the brackets are not properly balanced.

8. Conclusion

Provide the following:

• Summary of lessons learned

In this activity, I learned how to implement the stack data structure using various methods, including C++ STL, arrays, and linked lists. I also gained hands-on experience with different stack operations, such as push, pop, and empty, and the use of stacks, such as checking balanced symbols in expressions. Understanding how to work with stack

abstract data types (ADT) in multiple formats deepened my comprehension of their practical applications in problem-solving.

• Analysis of the procedure

The procedure followed a logical structure, starting with a basic introduction to stacks and their operations. Implementing the stack using both arrays and linked lists provided insight into the pros and cons of each approach, such as memory efficiency in arrays versus flexibility in linked lists. Testing the code through standard operations (push, pop, top, etc.) gave a solid foundation to compare their performance.

• Analysis of the supplementary activity

The supplementary activity, which involved checking balanced symbols in expressions, highlighted the effectiveness of stacks in parsing tasks. The implementation worked well for typical expressions, showcasing how LIFO order is advantageous in matching opening and closing symbols. The step-by-step approach, using stacks to solve the balancing problem, reinforced the theoretical concept of stacks by applying it to a practical problem.

• Concluding statement / Feedback: How well did you think you did in this activity? What are your areas for improvement?

I believe I did well in this activity. The implementation was successfully completed, and the stack operations functioned as expected. The supplementary activity demonstrated how stacks can be used in parsing algorithms. However, there is room for improvement in optimizing the implementation, particularly in ensuring edge cases are handled, such as when the stack is empty or when symbols don't match. Focusing on further improving my debugging skills and handling errors gracefully in complex expressions would be beneficial.

9. Assessment Rubric