

Activity No. 4.1	
Hands-on Activity 4.1 Stacks	
Course Code: CPE010	Program: Computer Engineering
Course Title: Data Structures and Algorithms	Date Performed: 04 / 10 / 24
Section: CPE21S4	Date Submitted: 04 / 10 / 24
Name(s): Leoj Jeam B. Tandayu	Instructor: Ms. Ma. Rizette Sayo

## 6. Output

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

It demonstrates the basic operations of a stack data structure, including pushing elements onto the stack, checking if the stack is empty, retrieving the top element, and popping elements off the stack.

Table 4-1. Output of ILO A

```
Enter number of max elements for new stack: 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty
1
New Value:
4
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty
1
New Value:
6
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty
3
The element on the top of the stack is 6
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEmpty
```

push : Inserts a new element on the top of the stack

pop : Removes the most recent element in the stack

Top : Displays the element at the top of the stack

isEmpty : Checks if the stack is empty

display : Outputs the element inside the stack

Table 4-2. Output of ILO B.1

```
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 :Stack is Empty.
Stack Underflow.
The Stack is Empty.
```

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

```
#include <iostream>
#include <stack>
#include <string>

using namespace std;

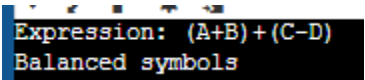
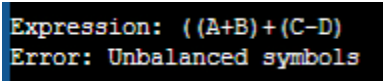
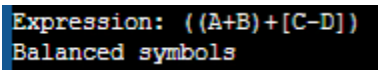
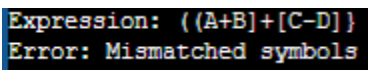
void checkBalancing(string expression) {
    stack<char> s;
    for (char c : expression) {
        if (c == '(') s.push('(');
        else if (c == '{') s.push('{');
        else if (c == '[') s.push('[');
        else if (c == ')' || c == '}' || c == ']') {
            if (s.empty()) {
                cout << "Error: Closing symbol without matching opening symbol" << endl;
                return;
            }
            char top = s.top();
            s.pop();
            if ((c == ')' && top != '(') || (c == '}' && top != '{') || (c == ']' && top != '[')) {
                cout << "Error: Mismatched symbols" << endl;
                return;
            }
        }
    }
}
```

```

    }
    if (!s.empty()) {
        cout << "Error: Unbalanced symbols" << endl;
        return;
    }
    cout << "Balanced symbols" << endl;
}

int main() {
    string expressions[] = {"(A+B)+(C-D)", "((A+B)+(C-D)", "((A+B)+[C-D])", "((A+B)+[C-D])"};
    for (string expression : expressions) {
        cout << "Expression: " << expression << endl;
        checkBalancing(expression);
        cout << endl;
    }
    return 0;
}

```

Expression	Valid? (Y/N)	Output (Console Screenshot)	Analysis
(A+B)+(C-D)	Y		It is balanced since both the opening and closing symbol is the same.
((A+B)+(C-D)	N		It is unbalanced since only the opening has a symbol and there is no closing symbol.
((A+B)+[C-D])	Y		It is balanced even if the 2 parts have different symbols. It is the outer opening and closing symbol that matters.
((A+B)+[C-D])	N		It is mismatched because the opening and closing have different symbols.

## 8. Conclusion

In conclusion, I have learned to apply my previously attained knowledge such as loops in stacks to continuously run the code and modify the contents of the stacks such as pop, top, and push. With this, I am able to understand more how stacks work.

## 9. Assessment Rubric