Activity No. 2	
Hands-on Activity 4.1 Stacks	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** October 04, 2024
Section: CPE21S4	**Date Submitted:** October 05, 2024
Name(s): Don Eleazar T. Fernandez	**Instructor:** Maria Rizette Sayo

6. Output

```cpp
//Tests the push, empty, size, pop, and top methods of the stack library.
#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;
}
```

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


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

Description: The stack was added with the values 3, 8, and 15. It then checked if the stack was empty using the .empty() function. After confirming the state of the stack, it displayed the number of items currently stored. Next, the .top() function was used to show the value at the top of the stack. Then, an element was removed using the .pop() function, and the top value was reevaluated, along with the updated number of remaining items.

Table 4 - 1. Output of ILO A

```cpp
1   #include<iostream>
2   const size_t maxCap= 100;
3   int stack[maxCap]; //stack with max of 100 elements
4   int top = -1, i, newData;
5   void push();
6   void pop();
7   void Top();
8   bool isEmpty();
9   void Display();
10  int main(){
11  int choice;
12  std::cout << "Enter number of max elements for new stack: ";
13  std::cin >> i;
14  while(true){
15  std::cout << "Stack Operations: " << std::endl;
16  std::cout << "1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY" << std::endl;
17  std::cin >> choice;
18  switch(choice){
19  case 1: push();
20  break;
21  case 2: pop();
22  break;
23  case 3: Top();
24  break;
25  case 4: std::cout << isEmpty() << std::endl;
26  break;
27  case 5: Display();
28  break;
29  default: std::cout << "Invalid Choice." << std::endl;
30  break;
31  }
32  }
33  return 0;
34  }
35  bool isEmpty(){
36  if(top==-1) return true;
37  return false;
38  }
39  void push(){
40  //check if full -> if yes, return error
41  if(top == i-1){
42  std::cout << "Stack Overflow." << std::endl;
43  return;
44  }
45  std::cout << "New Value: " << std::endl;
```

```cpp
46    std::cin >> newData;
47    stack[++top] = newData;
48    }
49    void pop(){
50    //check if empty -> if yes, return error
51    if(isEmpty()){
52    std::cout << "Stack Underflow." << std::endl;
53    return;
54    }
55    //display the top value
56    std::cout << "Popping: " << stack[top];
57    //decrement top value from stack
58    top--;
59    }
60    void Top(){
61    if(isEmpty()) {
62    std::cout << "Stack is Empty." << std::endl;
63    return;
64    }
65    std::cout << "The element on the top of the stack is " << stack[top] <<
66    std::endl;
67    }
68    void Display(){
69        if(isEmpty()) {
70            std::cout << "Stack is Empty." << std::endl;
71            return;
72        }
73        std::cout << "Stack elements: ";
74        for(int j = top; j >= 0; j--) {
75            std::cout << stack[j] << " ";
76        }
77        std::cout << std::endl;
78
79    }
```

```
Enter number of max elements for new stack: 2
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
1
New Value:
4
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
1
New Value:
8
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
5
Stack elements: 8 4
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
3
The element on the top of the stack is 8
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
2
Popping: 8Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
4
0
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
```

Description: The stack was initialized with a maximum capacity of two elements. The values four and eight were pushed into the stack. Operations included displaying the elements, getting the top value (which was eight, using the .top() function), removing an element with the .pop() function, and checking if the stack was empty using the .isEmpty() function. After reaching its maximum capacity, a value was popped from the stack. The output reflects the current state of the stack after it was emptied.

Table 4 - 2. Output of ILO B.1

```cpp
1  #include<iostream>
2  class Node{ public:
3  int data; Node *next;
4  };
5  Node *head=NULL,*tail=NULL; void push(int newData){
6  Node *newNode = new Node; newNode->data = newData; newNode->next = head;
7  if(head==NULL){
8  head = tail = newNode;
9  } else {
10 newNode->next = head; head = newNode;
11 }
12 }
13 int pop(){
14 int tempVal;
15 Node *temp;
16 if(head == NULL){
17 head = tail = NULL;
18 std::cout << "Stack Underflow." << std::endl; return -1;
19 } else {
20 temp = head;
21 tempVal = temp->data; head = head->next; delete(temp);
22 return tempVal;
23 }
24 }
25 void Top(){
26 if(head==NULL){
27 std::cout << "Stack is Empty." << std::endl; return;
28 } else {
29 std::cout << "Top of Stack: " << head->data << std::endl;
30 }
31 }
32 void display() {
33 if (head == NULL) {
34 std::cout << "Stack is Empty." << std::endl;
35 return;
36 } else {
37 Node* temp = head;
38 std::cout << "Stack elements: ";
39 while (temp != NULL) {
40 std::cout << temp->data << " ";
41 temp = temp->next;
42 }
43 std::cout << std::endl;
44 }
45 }
46 int main(){
47 push(1);
48 std::cout<<"After the first PUSH top of stack is :"; Top(); display();
49 push(5);
50 std::cout<<"After the second PUSH top of stack is :"; Top(); display();
51 pop();
52 std::cout<<"After the first POP operation, top of stack is:"; Top(); display();
53 pop();
54 std::cout<<"After the second POP operation, top of stack :"; Top(); display();
55 pop();
56 return 0;
57 }
```

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

Description: The stack works with simple .push() and .pop() functions. First, the value one is added to the stack, making it the top item. Then, the value five is added, which becomes the top. And when five is removed, one goes

back to the top. After removing one, the stack is empty. The message "Stack Underflow" was displayed because nothing could be removed from an empty stack. This demonstrates the concept of the stack, which is last in first out.

Table 4 - 3. Output of ILO B.2

7. Supplementary Activity

```cpp
#include <iostream>
#include <stack>
#include <string>
using namespace std;

bool isPair(char opening, char closing) {
    if (opening == '(' && closing == ')') return true;
    if (opening == '{' && closing == '}') return true;
    if (opening == '[' && closing == ']') return true;
    return false;
}

bool check(string expression) {
    stack<char> stack;
    for (char symbol : expression) {
        if (symbol == '(' || symbol == '{' || symbol == '[') {
            stack.push(symbol);
        } else if (symbol == ')' || symbol == '}' || symbol == ']') {
            if (stack.empty()) {
                cout << "Error: Closing Symbol Without A Match of Opening Symbol" << endl;
                return false;
            }
            char opening = stack.top();
            stack.pop();
            if (!isPair(opening, symbol)) {
                cout << "Error: Not Matched of Symbols" << endl;
                return false;
            }
        }
    }
    if (!stack.empty()) {
        cout << "Error: Not Balanced Symbols" << endl;
        return false;
    }
    return true;
}

int main() {
    string expression = "((A+B]+[C-D]}";
    if (check(expression)) {
        cout << "Balanced Symbols" << endl;
    } else {
        cout << "Not Balanced Symbols" << endl;
```
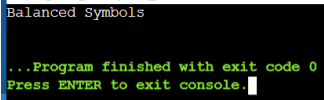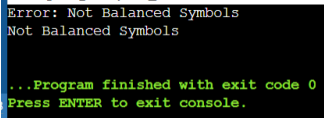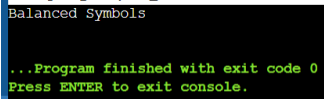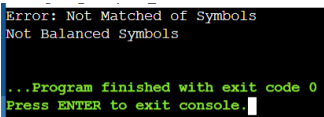
```
    }
    return 0;
}
```

Expression	Valid? (Y/N)	Output (Console Screenshot)	Analysis
(A+B)+(C-D)	Y	Balanced Symbols ...Program finished with exit code 0 Press ENTER to exit console.■	The output had resulted in "Balanced Symbols", as the expression had equal opening and closing symbols.
((A+B)+(C-D)	N	Error: Not Balanced Symbols Not Balanced Symbols ...Program finished with exit code 0 Press ENTER to exit console.	The output had resulted in "Not Balanced Symbols", as the expression had an extra opening symbol.
((A+B)+[C-D])	Y	Balanced Symbols ...Program finished with exit code 0 Press ENTER to exit console.	The output had resulted in "Balanced Symbols", as the expression had a pair of equal opening and closing symbols.
((A+B]+[C-D]}	N	Error: Not Matched of Symbols Not Balanced Symbols ...Program finished with exit code 0 Press ENTER to exit console.■	The output had resulted in "Not Balanced Symbols" with an error of "Not Matched of Symbols", as the expression had an unequal or different pair opening and closing symbols.

Tools Analysis: How do the different internal representations affect the implementation and usage of the stack?
- While the code is simple to create and easy to debug when using a simple stack that only saves symbols, it is much better at identifying and repairing faults when symbols and their positions are stored.

8. Conclusion

In this activity, I have learned more about the concept of stack, which is last in first out. Through the procedure, it demonstrated the functions in which it could operate, such as push(), pop(), top(), and isEmpty(). Along the activity, the capability of stack is displayed to have extended to the implementation of arrays and linked lists. In my case, I think I did well enough on this activity. What I could do to improve is to try and experiment much more about the capability of stack.

9. Assessment Rubric

I affirm that I will not give or receive any unauthorized help on this activity/exam and that all work will be my own.