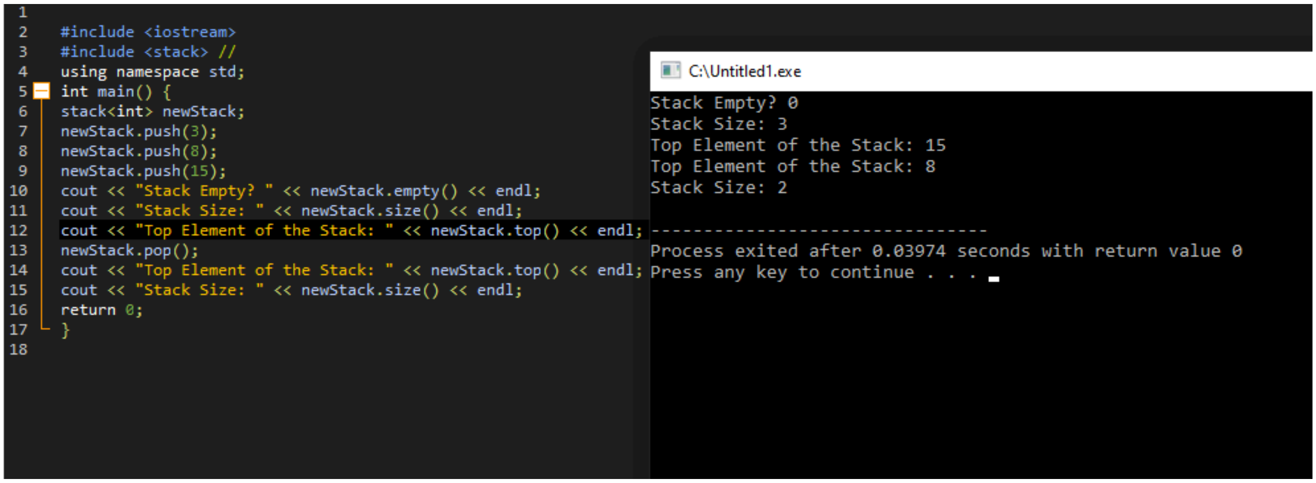
Activity No. 4

STACKS

Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:7/10/2024**
Section: CPE21S1	**Date Submitted: 7/10/2024**
Name(s): Jimenez, Christian Joros R.	**Instructor: Ms. Sayo**

6. Output

Table 4-1 Output ILO A

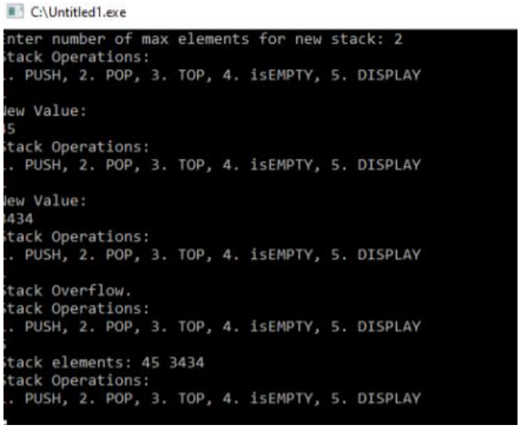


```cpp

#include <iostream>
#include <stack> //
using namespace std;
int main() {
stack<int> newStack;
newStack.push(3);
newStack.push(8);
newStack.push(15);
cout << "Stack Empty? " << newStack.empty() << endl;
cout << "Stack Size: " << newStack.size() << endl;
cout << "Top Element of the Stack: " << newStack.top() << endl;
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

--------------------------------
Process exited after 0.03974 seconds with return value 0
Press any key to continue . . .
```

Table 4-2 Output ILO B.1.

```cpp
#include<iostream>
const size_t maxCap= 100;
int stack[maxCap];
int top = -1, i, newData;
void push();
void pop();
void Top();
bool isEmpty();
void displayStack();
int main(){
int choice;
std::cout << "Enter number of max elements for new stack: ";
std::cin >> i;
while(true){
std::cout << "Stack Operations: " << std::endl;
std::cout << "1. PUSH, 2. POP, 3. TOP, 4. DISPLAY, 5. isEMPTY" << std::endl;
std::cin >> choice;
switch(choice){
case 1: push();
break;
case 2: pop();
break;
case 3: Top();
break;
case 4: displayStack();
break;
case 5: std::cout << isEmpty() << std::endl;
break;
default: std::cout << "Invalid Choice." << std::endl;
break;
}
}
return 0;
}
bool isEmpty(){
if(top==-1) return true;
return false;
}
void push(){
if(top == i-1){
std::cout << "Stack Overflow." << std::endl;
return;
}
std::cout << "New Value: " << std::endl;
std::cin >> newData;
stack[++top] = newData;                                        }
void pop(){
if(isEmpty()){
std::cout << "Stack Underflow." << std::endl;
return;
}
std::cout << "Popping: " << stack[top] << std::endl;
top--;
}
void Top(){
if(isEmpty()) {
std::cout << "Stack is Empty." << std::endl;
return;
}
std::cout << "The element on the top of the stack is " << stack[top] <<
std::endl;
}
void displayStack() {
if (isEmpty()) {
std::cout << "Stack is empty." << std::endl;
return;
}
std::cout << "Stack elements: ";
for (int j = top; j >= 0; j--) {
std::cout << stack[j] << " ";
}
std::cout << std::endl;
}
```



C:\Untitled1.exe

```
nter number of max elements for new stack: 2
tack Operations:
. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY

ew Value:
5
tack Operations:
. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY

ew Value:
434
tack Operations:
. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY

tack Overflow.
tack Operations:
. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY

tack elements: 45 3434
tack Operations:
. PUSH, 2. POP, 3. TOP, 4. isEMPTY, 5. DISPLAY
```

1. **maxCap:** This constant defines the maximum capacity of the stack (100 elements).
2. **stack[maxCap]:** This is an array of integers, representing the stack.
3. **top:** This variable keeps track of the index of the top element in the stack. It's initialized to -1, indicating an empty stack.
4. **i:** This variable is used to store the user-defined size of the stack.
5. **newData:** This variable is used to store the new data to be pushed onto the stack.
6. **push():**
   - Checks if the stack is full (top == i-1). If full, it prints "Stack Overflow" and returns.

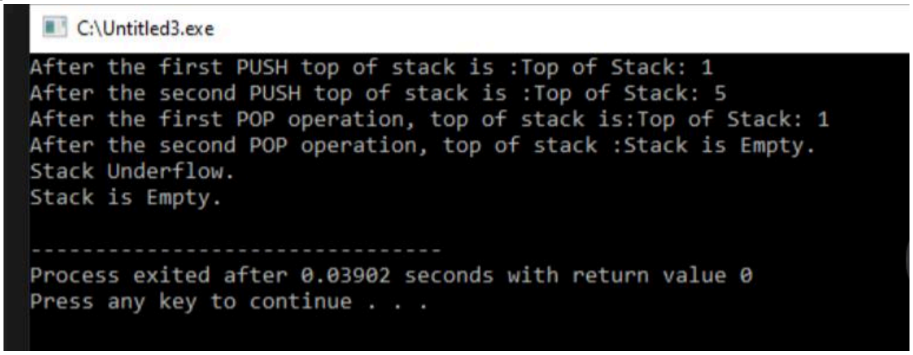
pop():
Checks if the stack is empty (isEmpty()). If empty, it prints "Stack Underflow" and returns.
Top():
Checks if the stack is empty. If empty, it prints "Stack is Empty" and returns.
isEmpty():
Returns true if the stack is empty (top == -1), otherwise returns false.

Table 4-2 Output ILO B.2.

```
#include<iostream>
class Node{
public:
int data;
Node *next;
};
Node *head=NULL,*tail=NULL;

void push(int newData){
Node *newNode = new Node;
newNode->data = newData;
newNode->next = head;
if(head==NULL){
head = tail = newNode;
} else {
newNode->next = head;
head = newNode;
}
}

int pop(){
int tempVal;
Node *temp;                                                        if(head == NULL){
```

```
head = tail = NULL;
std::cout << "Stack Underflow." << std::endl;
return -1;
} else {
temp = head;
tempVal = temp->data;
head = head->next;
delete(temp);
return tempVal;
}
}

void Top(){
if(head==NULL){
std::cout << "Stack is Empty." << std::endl;
return;
} else {
std::cout << "Top of Stack: " << head->data << std::endl;
}
}

void displayStack(){
if (head == NULL) {
std::cout << "Stack is empty." << std::endl;
return;
}
std::cout << "Stack elements: ";
Node *current = head;
while (current != NULL) {
std::cout << current->data << " ";
current = current->next;
}
std::cout << std::endl;
}

int main(){
push(1);
std::cout<<"After the first PUSH top of stack is :";
Top();
push(5);
std::cout<<"After the second PUSH top of stack is :";
Top();
pop();
std::cout<<"After the first POP operation, top of stack is:";
Top();
pop();
std::cout<<"After the second POP operation, top of stack :";
Top();
pop();

displayStack();
```

return 0;
}

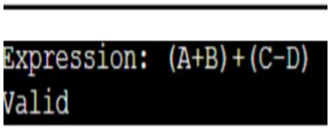
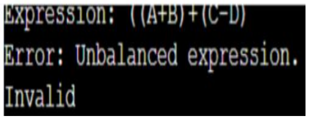
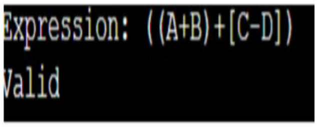


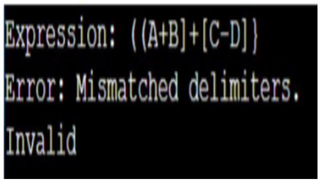
C:\Untitled3.exe

```
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.
Stack is Empty.

------------------------------
Process exited after 0.03902 seconds with return value 0
Press any key to continue . . .
```

Node class: Represents a node in the linked list.
data: Stores the data value of the node.
next: Points to the next node in the list (or NULL if it's the last node).
head: Pointer to the first node of the linked list (top of the stack).
tail: Pointer to the last node of the linked list (bottom of the stack).
Stack Operations:
push(int newData):
Creates a new Node with the given newData.
Sets the next pointer of the new node to the current head.

7. Supplementary Activity

Expression	Valid? (Y/N)	Output (Console Screenshot)	Analysis
(A+B)+(C-D)	Y	 Expression: (A+B)+(C-D) Valid	The symbols in the statement are balanced.the opening and closing delimiters, respectively, and are matched with them.
((A+B)+(C-D)	N	 Expression: ((A+B)+(C-D) Error: Unbalanced expression. Invalid	a closing delimiter is not present in this expression at the end, the stack is not empty.
((A+B)+[C-D])	Y	 Expression: ((A+B)+[C-D]) Valid	this expression has balanced symbols

((A+B]+[C-D]}	N	 Expression: ((A+B]+[C-D]} Error: Mismatched delimiters. Invalid	this statement had mismatched delimiters. the opening delimiter shouldnt be matched with the closing delimiter.

8. Conclusion

• Summary of lessons learned
i learned how to utilize c++ stl and create stacks using it, i was also able to create a c++ program that uses both arrays and linked lists for stacks.
• Analysis of the procedure
I was able to solve the problems of this activity by understanding the problem and using my knowledge to solve these problems.
• Analysis of the supplementary activity
the supplementary activity was a bit harder to understand because it was quite unfamiliar, but with the help of googling and analysis, i was able to answer the supplementary activity
• Concluding statement / Feedback: How well did you think you did in this activity? What are your areas
for improvement?
i think i did okay in this activity, although i was a bit rushed in doing thsi activity because the deadline was getting close.

9. Assessment Rubric