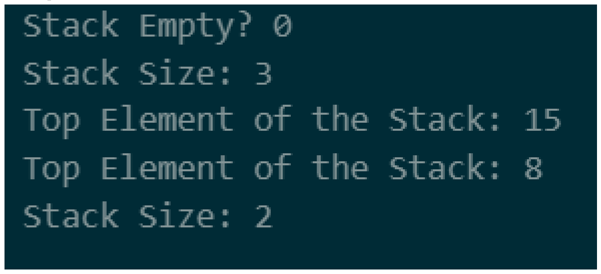
Activity No. 4	
STACKS	
Course Code: CPE010	**Program:** Computer Engineering
Course Title: Data Structures and Algorithms	**Date Performed:** Aug 26, 2025
Section: CPE21S4	**Date Submitted:** Aug 26, 2025
Name(s): Punay, Heidee S.	**Instructor:** Engr. Jimlord Quejado
6. Output	

Procedure:

A: Create a stack using the C++ STL

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

int main()
{
    std::stack<int>newStack;

    newStack.push(3);
    newStack.push(8);
    newStack.push(15);

    std::cout<<"Stack Empty? "<< newStack.empty()<<std::endl;
    std::cout<<"Stack Size: " << newStack.size()<<std::endl;
    std::cout<<"Top Element of the Stack: "<<newStack.top()<<std::endl;

    newStack.pop();

    std::cout<<"Top Element of the Stack: "<<newStack.top()<<std::endl;
    std::cout<<"Stack Size: "<<newStack.size()<<std::endl;

    return 0;
}
```

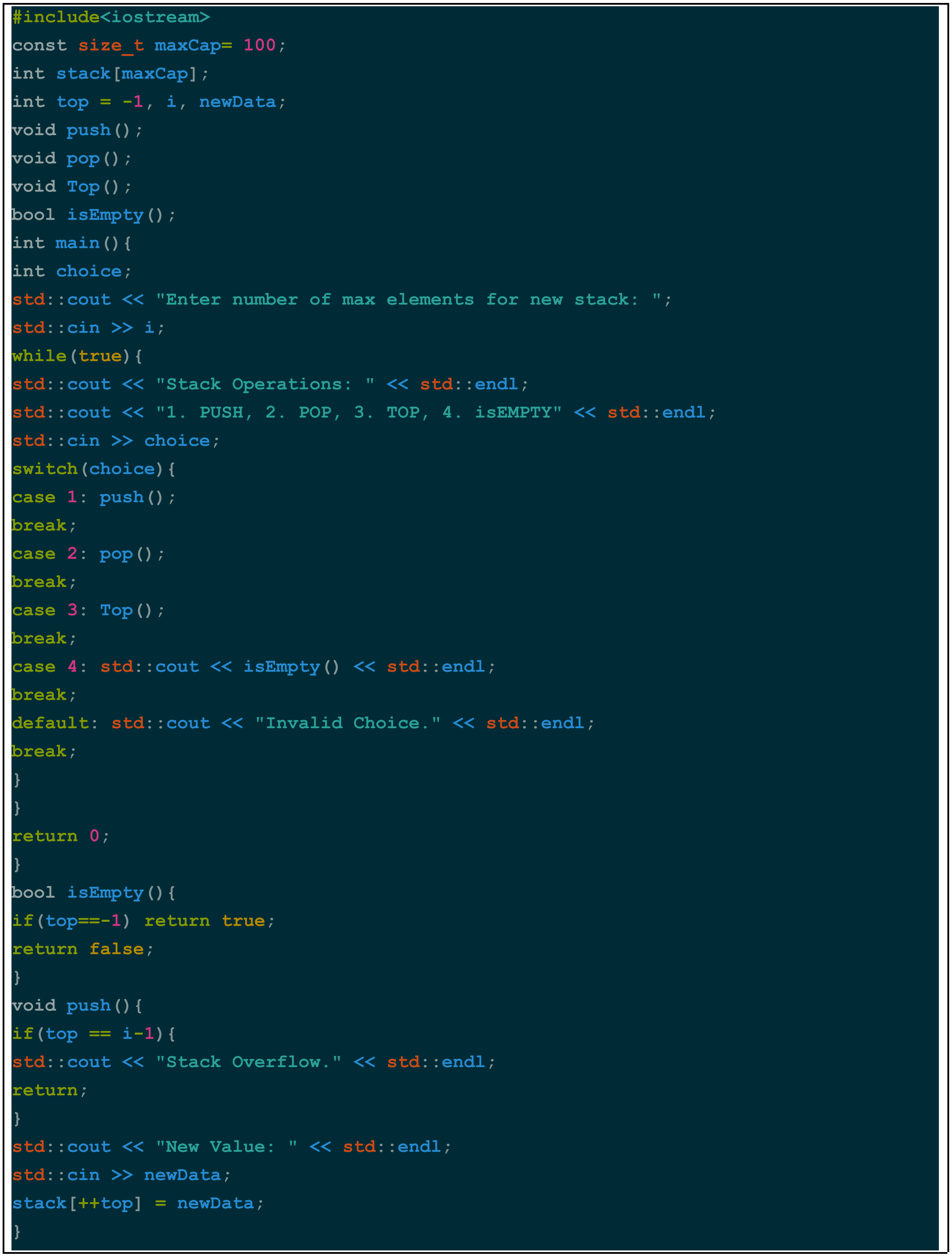
Output:



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

3, 8, and 15 were the elements pushed in a stack where it is now has a size of 3. The last element that is push will be the top element where it is popped and the size becomes 2 and has a new of top element which is the 8.

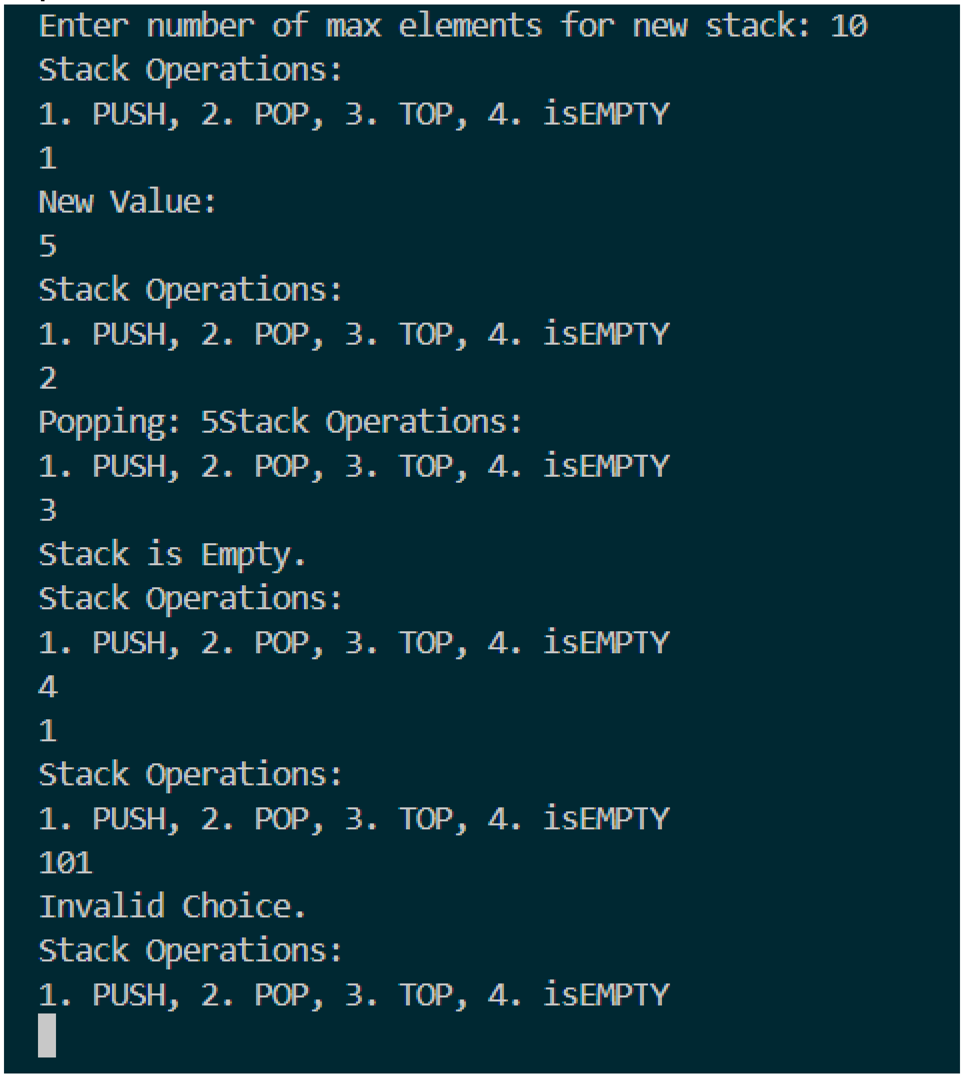
B.1. Stacks using Arrays



```cpp
#include<iostream>
const size_t maxCap= 100;
int stack[maxCap];
int top = -1, i, newData;
void push();
void pop();
void Top();
bool isEmpty();
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. isEMPTY" << std::endl;
std::cin >> choice;
switch(choice){
case 1: push();
break;
case 2: pop();
break;
case 3: Top();
break;
case 4: 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;
}
```

```cpp
void pop(){
if(isEmpty()){
std::cout << "Stack Underflow." << std::endl;
return;
}
std::cout << "Popping: " << stack[top];
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;
}
```

Output:



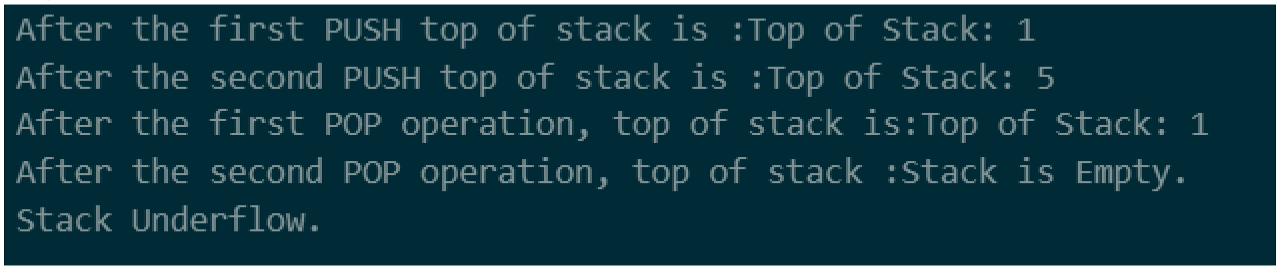
```
Enter number of max elements for new stack: 10
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
1
New Value:
5
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
2
Popping: 5Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
3
Stack is Empty.
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
4
1
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
101
Invalid Choice.
Stack Operations:
1. PUSH, 2. POP, 3. TOP, 4. isEMPTY
```

The switch case inside the while function, which tells that if the bool is empty, then it goes to the while function because it's true. Then, in there, we choose a stack operation and input numbers that will be put in a stack.

B.2. Stacks using Linked Lists

```
#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;
}
}
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();
return 0;
}
```

Output:



```
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.
```

7. Supplementary Activity

8. Conclusion

To conclude, I was able to learn more about the other functions that can be used in stacks. Though linked lists it is very hard to understand and apply, or even analyze the code. Overall, there are many functions that can be used in creating a stack, like an array or the STL, which is more way easier to understand than linked list.

9. Assessment Rubric