Department of Electronics & Telecommunication Engineering
Data Structures & Algorithms Lab (DJ19ECSBL1)

Experiment no: 1 **Date: 22-09-2022**

Name: Nihal Nisar Shaikh **SAP ID: 60002200155** **Batch: E1-4**

Aim: Write a menu driven program in C to implement stack

Write a program in C to reverse a string using stack.

Software Language: C

Theory: A stack is a linear data structure that follows the principle of **Last In First Out (LIFO)**. This means the last element inserted inside the stack is removed first.

Basic Operations of Stack:

There are some basic operations that allow us to perform different actions on a stack.

- Push: Add an element to the top of a stack
  - IsFull: Check if the stack is full(overflow condition)
- Pop: Remove an element from the top of a stack
  - IsEmpty: Check if the stack is empty(underflow condition)
- Peek: Get the value of the top element without removing it

Algorithm for push operation:

Step 1 − Checks if the stack is full.

Step 2 − If the stack is full, then display “overflow” and exit.

Step 3 − If the stack is not full, increments top to point next empty space.

Step 4 − Adds data element to the stack location, where top is pointing.

Step 5 − Returns success.

Algorithm for pop operation:

Step 1 − Checks if the stack is empty.

Step 2 − If the stack is empty, then display “underflow” and exit.

Step 3 − If the stack is not empty, access the data element at which top is pointing.

Step 4 − Decrease the value of top by 1.

Step 5 − Returns success.

Algorithm for peek operation:

Step 1 - Check whether stack is EMPTY. (top == -1)

Step 2 - If it is EMPTY, then display "Stack is EMPTY!!!" and terminate the function.

Step 3 - If it is NOT EMPTY, then display top element of the stack.

Step 4 - Return success

Algorithm to reverse a string:

Step 1 – Create an empty stack.

Step 2 - Pick the characters from the string one by one and put them to the stack, so that the last character of the string comes at the top of the stack.

Step 3 - Pop the stack and put the popped characters back in the empty string.

Procedure of menu driven program:

Step 1 - Include all the header files which are used in the program and define a constant 'SIZE' with specific value.

Step 2 - Declare all the functions used in stack implementation.

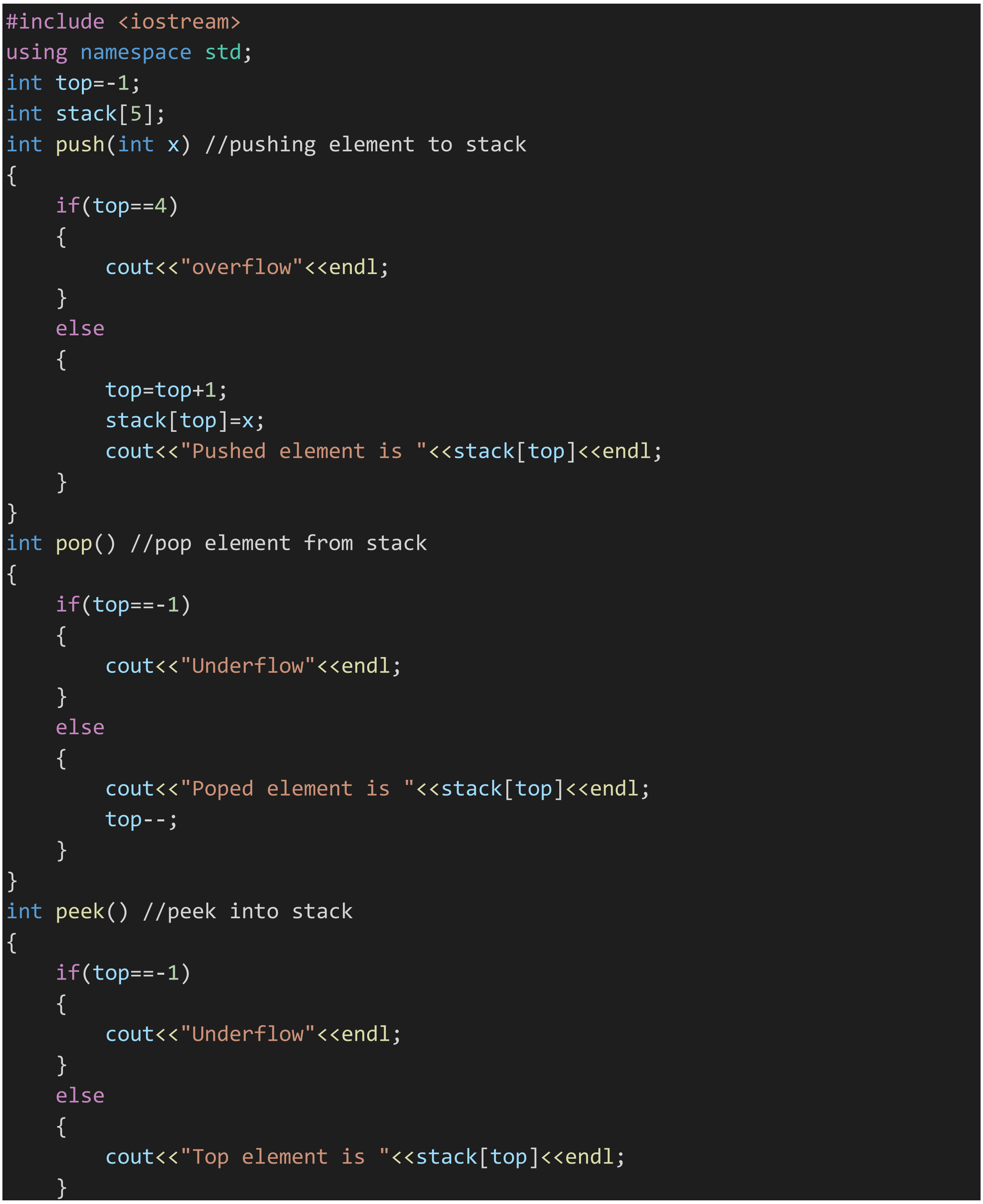
Step 3 - Create a one dimensional array with fixed size (int stack[SIZE])

Step 4 - Define an integer variable 'top' and initialize with '-1'. (int top = -1)

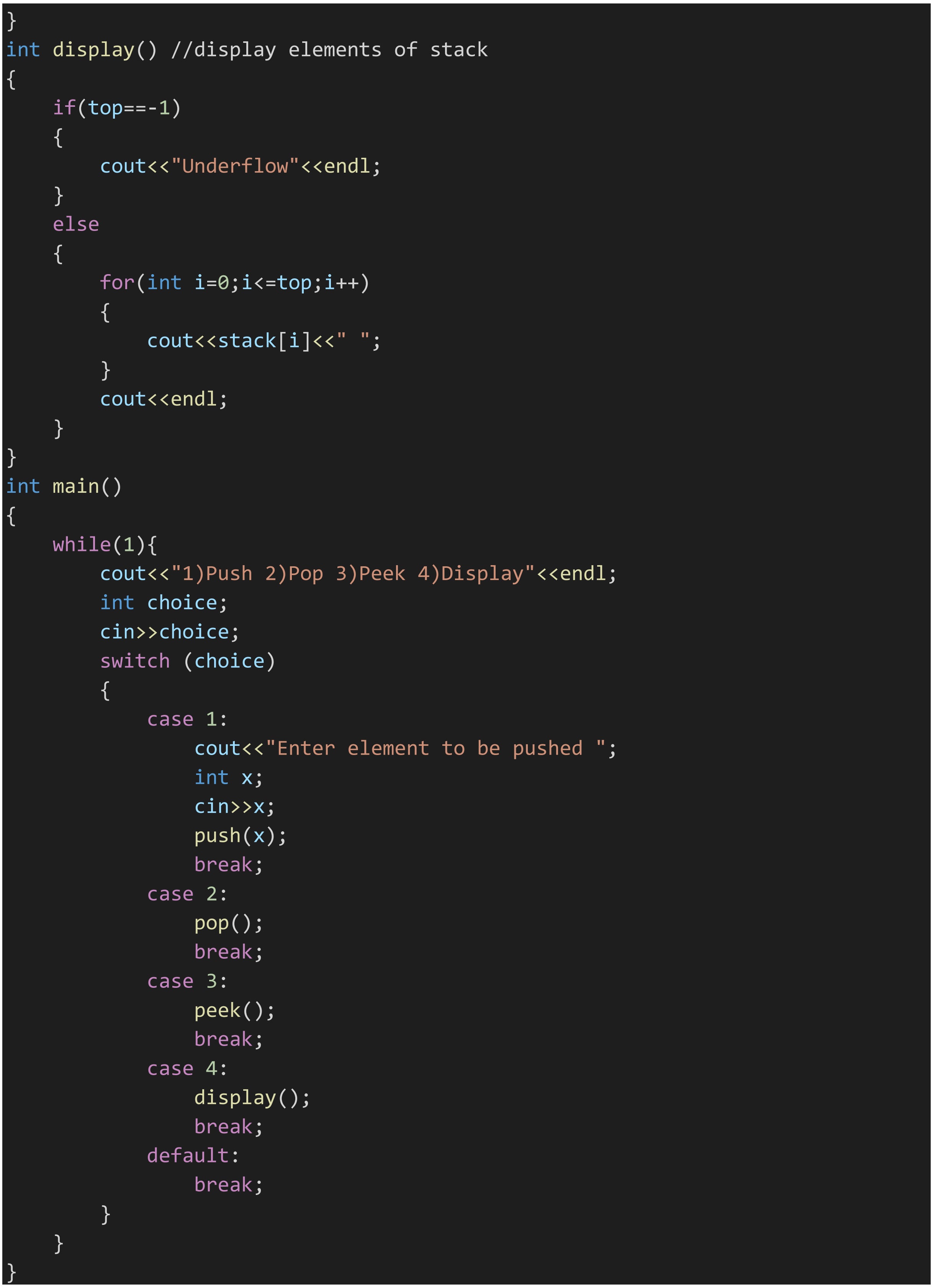
Step 5 - In main method, display menu with list of operations and make suitable function calls to perform operation selected by the user on the stack.

Code:

1) **Basic Operations of Stack:**



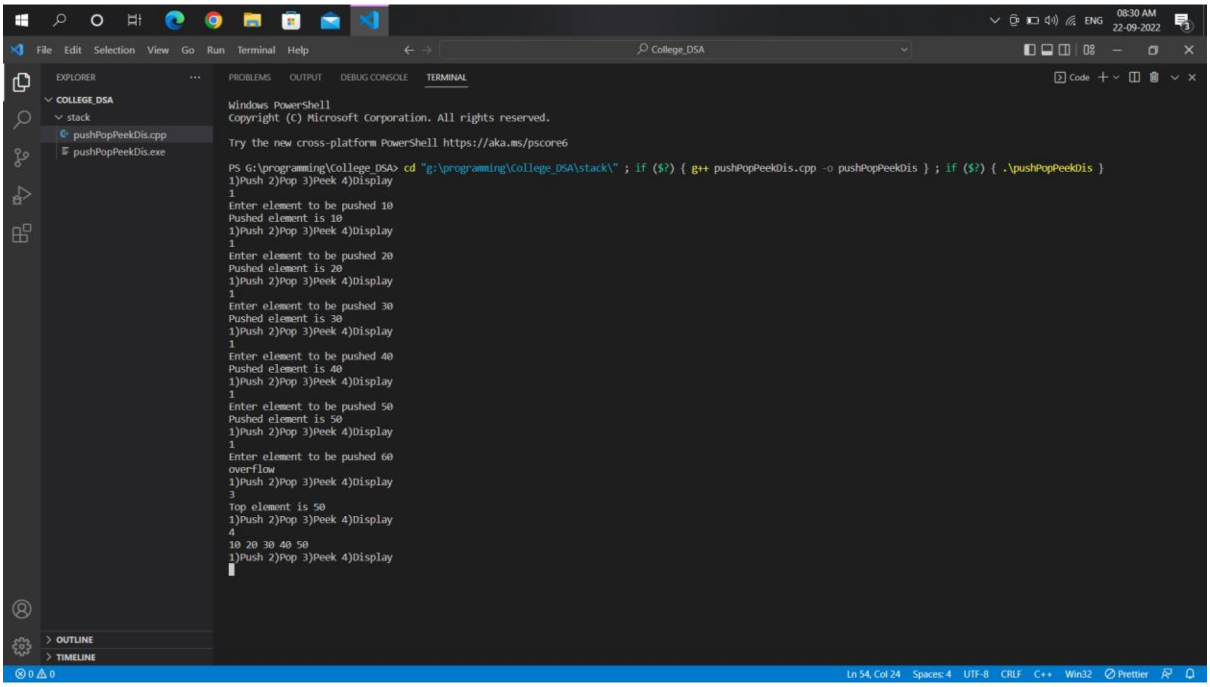
```
#include <iostream>
using namespace std;
int top=-1;
int stack[5];
int push(int x) //pushing element to stack
{
    if(top==4)
    {
        cout<<"overflow"<<endl;
    }
    else
    {
        top=top+1;
        stack[top]=x;
        cout<<"Pushed element is "<<stack[top]<<endl;
    }
}
int pop() //pop element from stack
{
    if(top==-1)
    {
        cout<<"Underflow"<<endl;
    }
    else
    {
        cout<<"Poped element is "<<stack[top]<<endl;
        top--;
    }
}
int peek() //peek into stack
{
    if(top==-1)
    {
        cout<<"Underflow"<<endl;
    }
    else
    {
        cout<<"Top element is "<<stack[top]<<endl;
    }
```



```
}
int display() //display elements of stack
{
    if(top==-1)
    {
        cout<<"Underflow"<<endl;
    }
    else
    {
        for(int i=0;i<=top;i++)
        {
            cout<<stack[i]<<" ";
        }
        cout<<endl;
    }
}
int main()
{
    while(1){
        cout<<"1)Push 2)Pop 3)Peek 4)Display"<<endl;
        int choice;
        cin>>choice;
        switch (choice)
        {
            case 1:
                cout<<"Enter element to be pushed ";
                int x;
                cin>>x;
                push(x);
                break;
            case 2:
                pop();
                break;
            case 3:
                peek();
                break;
            case 4:
                display();
                break;
            default:
                break;
        }
    }
}
```

Outputs:

1) Pushing elements to stack:
i)Pushing -ii)Checking overflow condition

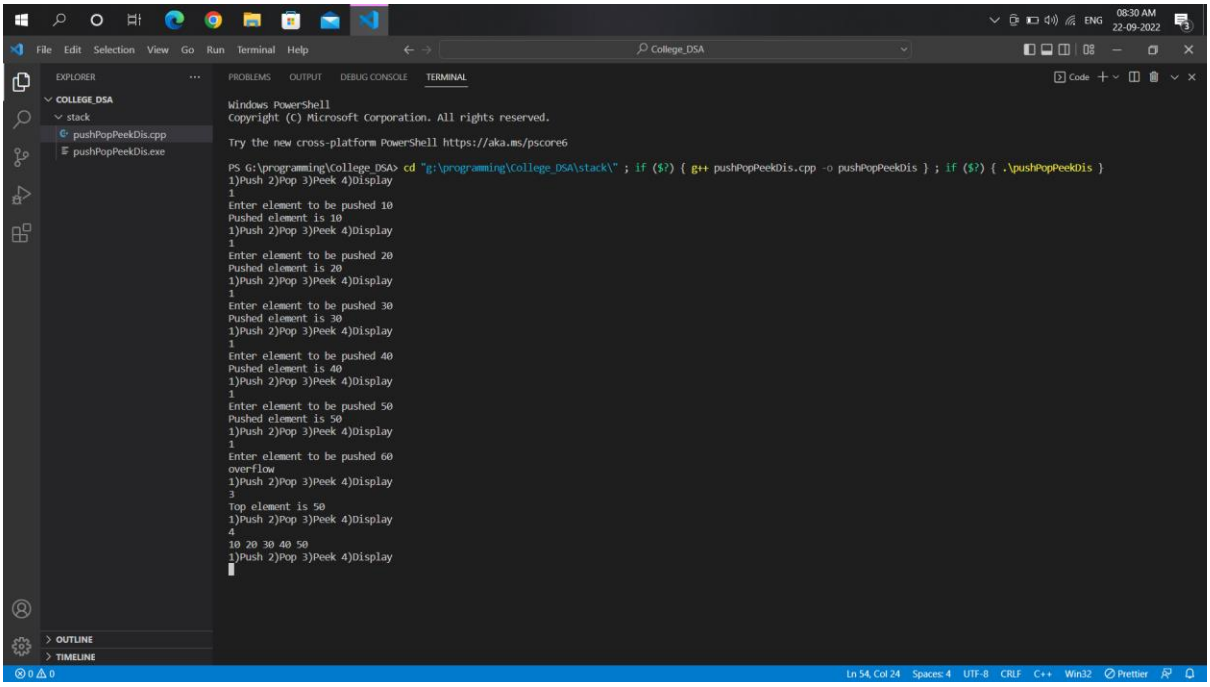


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS G:\programming\College_DSA> cd "g:\programming\College_DSA\stack\" ; if ($?) { g++ pushPopPeekDis.cpp -o pushPopPeekDis } ; if ($?) { .\pushPopPeekDis }
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 10
Pushed element is 10
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 20
Pushed element is 20
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 30
Pushed element is 30
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 40
Pushed element is 40
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 50
Pushed element is 50
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 60
overflow
1)Push 2)Pop 3)Peek 4)Display
3
Top element is 50
1)Push 2)Pop 3)Peek 4)Display
4
10 20 30 40 50
1)Push 2)Pop 3)Peek 4)Display
```

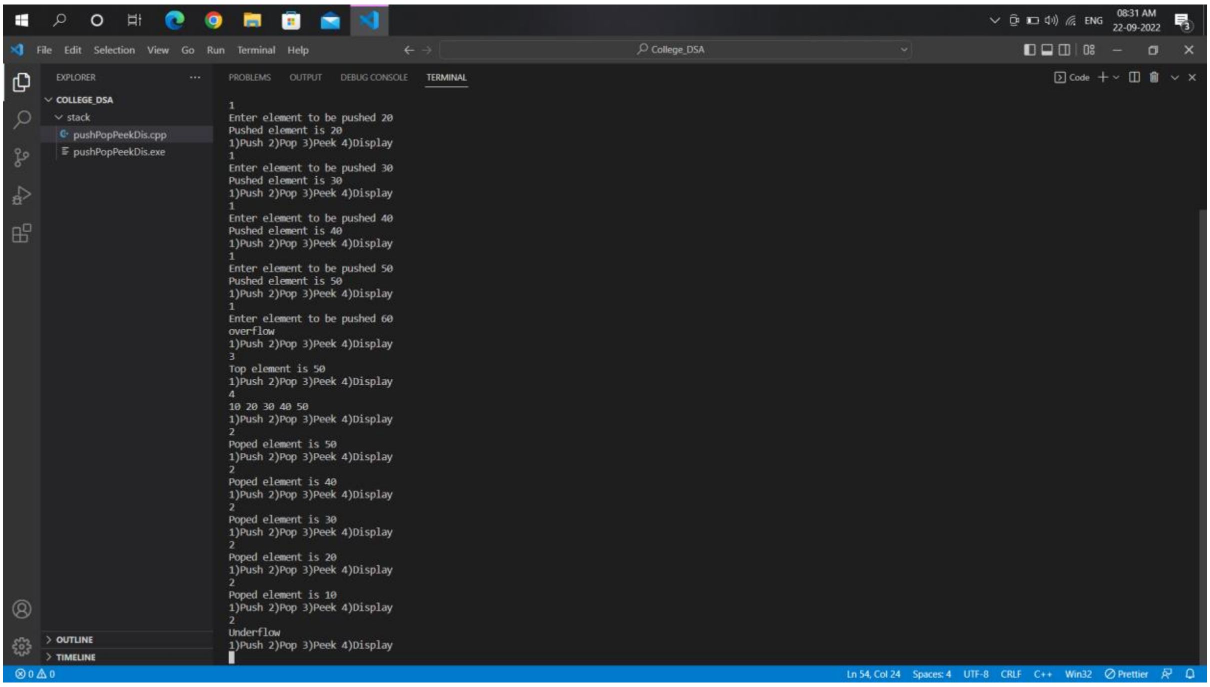
2) Peek and Display:


Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

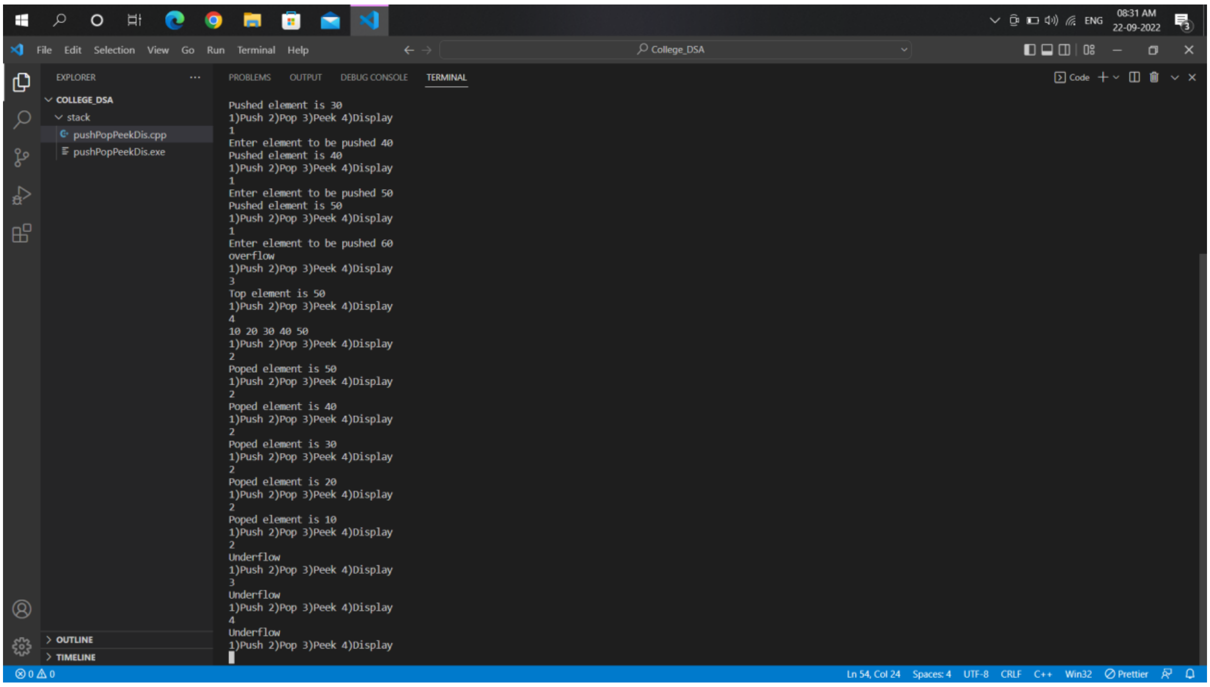
PS G:\programming\College_DSA> cd "g:\programming\College_DSA\stack\" ; if ($?) { g++ pushPopPeekDis.cpp -o pushPopPeekDis } ; if ($?) { .\pushPopPeekDis }
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 10
Pushed element is 10
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 20
Pushed element is 20
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 30
Pushed element is 30
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 40
Pushed element is 40
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 50
Pushed element is 50
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 60
overflow
1)Push 2)Pop 3)Peek 4)Display
3
Top element is 50
1)Push 2)Pop 3)Peek 4)Display
4
10 20 30 40 50
1)Push 2)Pop 3)Peek 4)Display

3) Pop elements from stack:



```
1
Enter element to be pushed 20
Pushed element is 20
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 30
Pushed element is 30
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 40
Pushed element is 40
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 50
Pushed element is 50
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 60
overflow
1)Push 2)Pop 3)Peek 4)Display
3
Top element is 50
1)Push 2)Pop 3)Peek 4)Display
4
10 20 30 40 50
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 50
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 40
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 30
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 20
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 10
1)Push 2)Pop 3)Peek 4)Display
2
Underflow
1)Push 2)Pop 3)Peek 4)Display
```

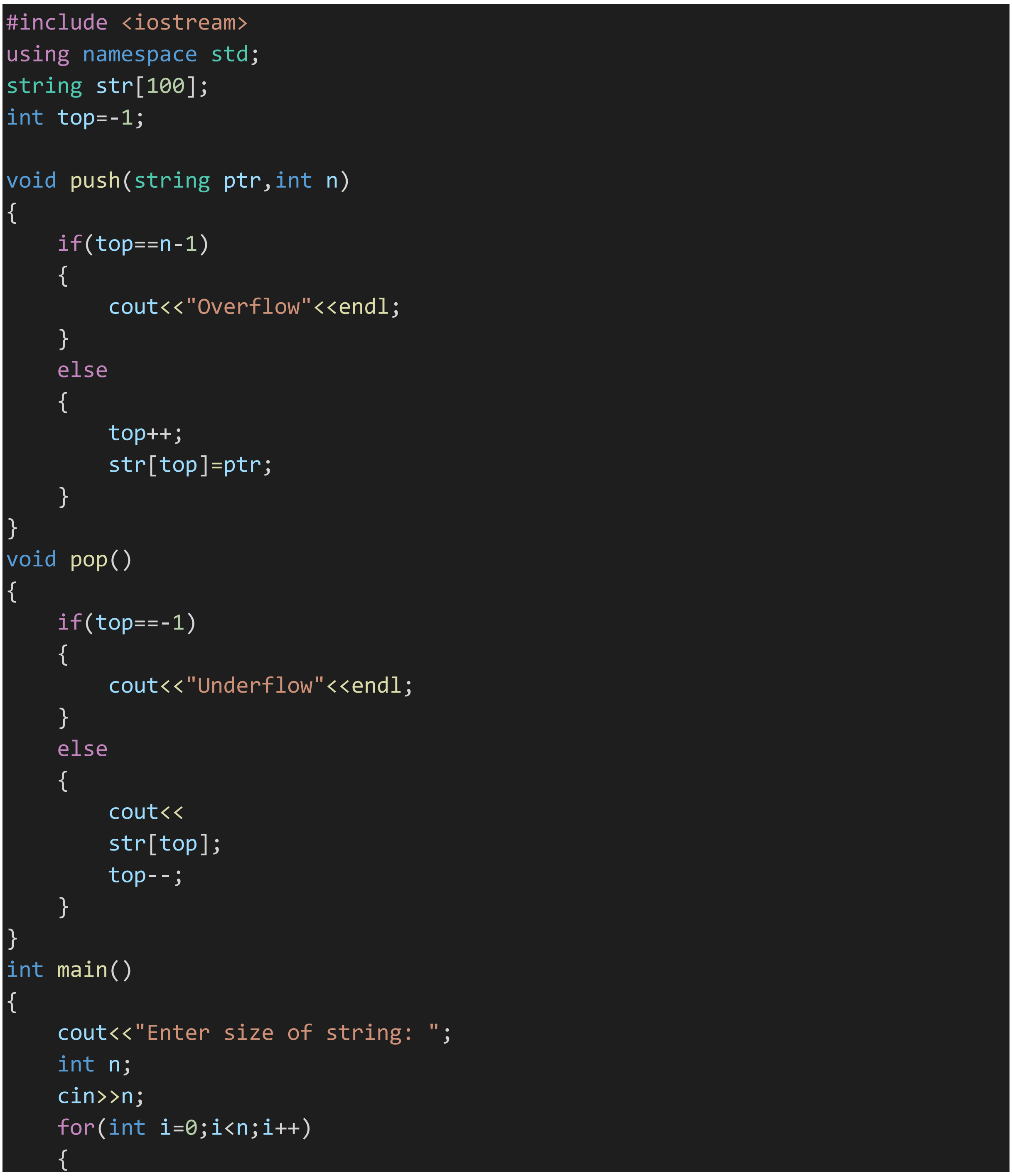
4) Checking underflow condition after popping all elements:



```
Pushed element is 30
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 40
Pushed element is 40
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 50
Pushed element is 50
1)Push 2)Pop 3)Peek 4)Display
1
Enter element to be pushed 60
overflow
1)Push 2)Pop 3)Peek 4)Display
3
Top element is 50
1)Push 2)Pop 3)Peek 4)Display
4
10 20 30 40 50
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 50
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 40
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 30
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 20
1)Push 2)Pop 3)Peek 4)Display
2
Poped element is 10
1)Push 2)Pop 3)Peek 4)Display
2
Underflow
1)Push 2)Pop 3)Peek 4)Display
3
Underflow
1)Push 2)Pop 3)Peek 4)Display
4
Underflow
1)Push 2)Pop 3)Peek 4)Display
```

Code:

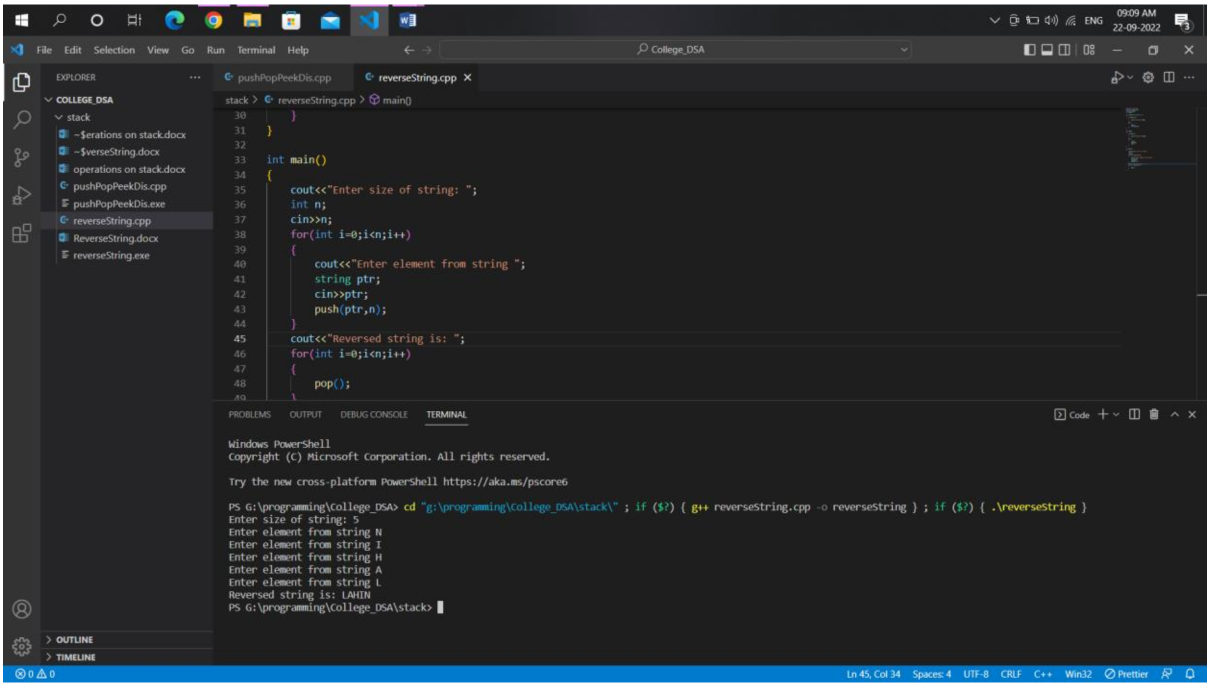
2) **Reverse a string:**



```
#include <iostream>
using namespace std;
string str[100];
int top=-1;

void push(string ptr,int n)
{
    if(top==n-1)
    {
        cout<<"Overflow"<<endl;
    }
    else
    {
        top++;
        str[top]=ptr;
    }
}
void pop()
{
    if(top==-1)
    {
        cout<<"Underflow"<<endl;
    }
    else
    {
        cout<<
        str[top];
        top--;
    }
}
int main()
{
    cout<<"Enter size of string: ";
    int n;
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cout<<"Enter element from string ";
        string ptr;
        cin>>ptr;
        push(ptr,n);
    }
    cout<<"Reversed string is: ";
    for(int i=0;i<n;i++)
    {
        pop();
    }
}
```

Output:



```
        }
}

int main()
{
    cout<<"Enter size of string: ";
    int n;
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cout<<"Enter element from string ";
        string ptr;
        cin>>ptr;
        push(ptr,n);
    }
    cout<<"Reversed string is: ";
    for(int i=0;i<n;i++)
    {
        pop();
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS G:\programming\College_DSA> cd "g:\programming\College_DSA\stack\" ; if ($?) { g++ reverseString.cpp -o reverseString } ; if ($?) { .\reverseString }
Enter size of string: 5
Enter element from string N
Enter element from string I
Enter element from string H
Enter element from string A
Enter element from string L
Reversed string is: LAHIN
PS G:\programming\College_DSA\stack>
```

Result and Conclusion:

1) **Stack is a simple linear data structure used for storing data.**
2) **Stack follows the LIFO(Last In First Out) strategy that states that the element that is inserted last will come out first.**
3) **It can be implemented through an array or linked lists.**
4) **Some of its main operations are: push(), pop(), top(), isEmpty(), size(), etc.**

In order to make manipulations in a stack, there are certain operations provided to us:

5) **When we want to insert an element into the stack the operation is known as the push operation whereas when we want to remove an element from the stack the operation is known as the pop operation. If we try to pop from an empty stack then it is known as underflow and if we try to push an element in a stack that is already full, then it is known as overflow.**

6) **Implementation of stack:**

   - Evaluation of Arithmetic Expressions
   - **Backtracking**
   - Delimiter Checking
   - **Reverse a Data**
   - Processing Function Calls