

Name: - Atish Kumar

Roll No: - 120CS0173

Lab Sheet: - 06

Q1. Write a program to implement the stack using an array.

Note: The code should have the modularity and should include following function apart from main ():

- **Push()** This function inserts an element to top of the Stack.
- **Pop()** This function deletes an element from top of the Stack.
- **Display()** This function displays all the elements of the Stack by popping them one by one.

Program:-

```
#include<iostream>
```

```
using namespace std;
```

```
int stack[100], n = 100, ptr = -1;
```

```
void push(int val){
```

```
    if(ptr>=n-1){
```

```
        cout<<"Stack Overflow"<<endl;
```

```
    }
```

```
    else{
```

```
        ptr++;
```

```
        stack[ptr] = val;
```

```
    }
```

```
}
```

```
void pop(){
```

```
    if(ptr<=-1){
```

```
        cout<<"Stack Underflow"<<endl;
```

```
    }
```

```
    else{
```

```
        ptr--;
```

```

    }
}

void top(){
    cout<<"The top element is "<<stack[ptr]<<endl;
}

void top_for_display(){
    cout<<stack[ptr]<<" ";
}

void display(){
    if(ptr<0){
        cout<<"Stack is empty!!";
        return;
    }
    cout<<"Stack elements are: ";
    while(ptr>=0){
        top_for_display();
        pop();
    }

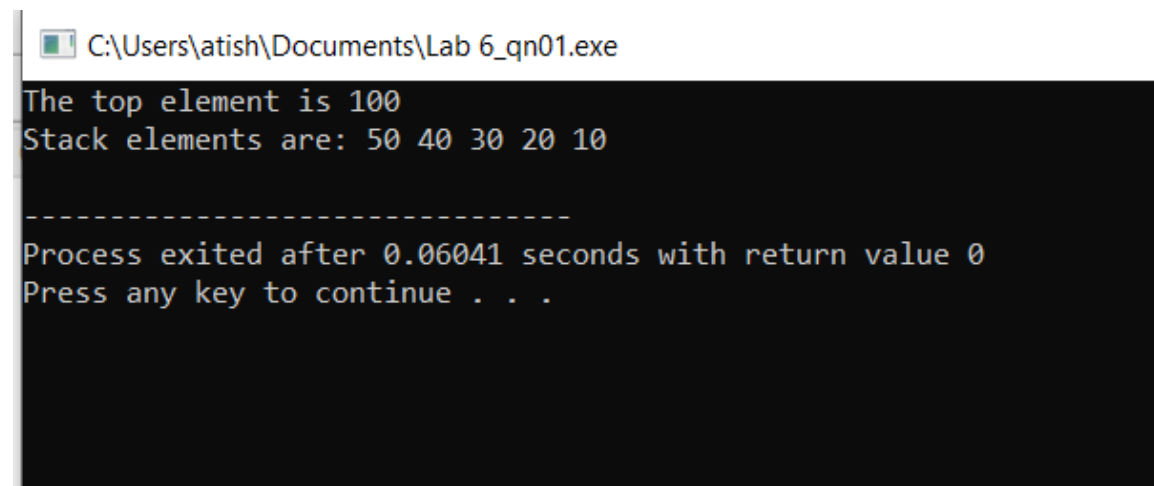
    cout<<endl;
}

int main(){
    push(10);
    push(20);
    push(30);
    push(40);
    push(50);

```

```
    push(60);  
    pop();  
    push(100);  
    top();  
    pop();  
    display();  
    return 0;  
}
```

Input:-



```
C:\Users\atish\Documents\Lab 6_qn01.exe  
The top element is 100  
Stack elements are: 50 40 30 20 10  
-----  
Process exited after 0.06041 seconds with return value 0  
Press any key to continue . . .
```

Q2. Convert an infix expression to its equivalent postfix expression.

Note: The code should have the modularity and should include following functions apart from `main()`:

- `getNextToken()`: This function returns the next token in the input infix expression. The token may be an operator or "(" or ")" or an operand. The operands can be of multiple digits. For example the infix expression `1000/(10+240)` contains operands 1000, 10, and 240.
- `infixToPostfix()`: Converts the input infix expression to postfix. This function calls `getNextToken()` repeatedly to get the next token until it reaches the end. The token is then processed depending on whether it is an operand or operator or (or).

Program:-

```
#include <iostream>

#include <stack>

#include <string.h>

#include <bits/stdc++.h>

using namespace std;

int last = 0;

int precedence(string s)
{
    if (s == "+" || s == "-")
        return 1;
    if (s == "*" || s == "/")
        return 2;
    if (s == "^")
        return 3;
    if (s == "(" || s == ")")
        return 0;
    return -1;
}

string getNextToken(string s)
```

```

{
    string s1 = "";
    int p = 0;
    if (last == s.length())
    {
        s1 = "-1";
        return s1;
    }
    for (int i = last; i < s.length(); i++)
    {
        if (s[i] >= '0' && s[i] <= '9')
        {
            s1 = s1 + s[i];
            last++;
            p = 1;
        }
        if (s[i] == '*' || s[i] == '/' || s[i] == '+' || s[i] == '-' || s[i] == '^' || s[i] == '('
|| s[i] == ')')
        {
            if (p == 0)
            {
                last++;
                s1 = s[i];
                return s1;
            }
            else
            {

```

```

        return s1;
    }
}

return s1;
}

string infixToPostfix(string s)
{
    stack<string> st;
    string postfix = "";
    string s1;
    while (last < s.length())
    {
        s1 = getNextToken(s);

        if(s1=="-1"){
            break;
        }
        if (precedence(s1) == -1){
            postfix = postfix + s1;

        }
        else if (precedence(s1) == 0)
        {
            cout<<s1<<endl;

```

```

if (s1 == "{"){

    st.push("{");

}
else if (s1 == ")")
{
    while (!st.empty() && st.top() != "{")
    {
        postfix = postfix + st.top();

        st.pop();
    }
    st.pop();
}
else
{
    while (!st.empty() && precedence(st.top()) >= precedence(s1))
    {
        postfix += st.top();

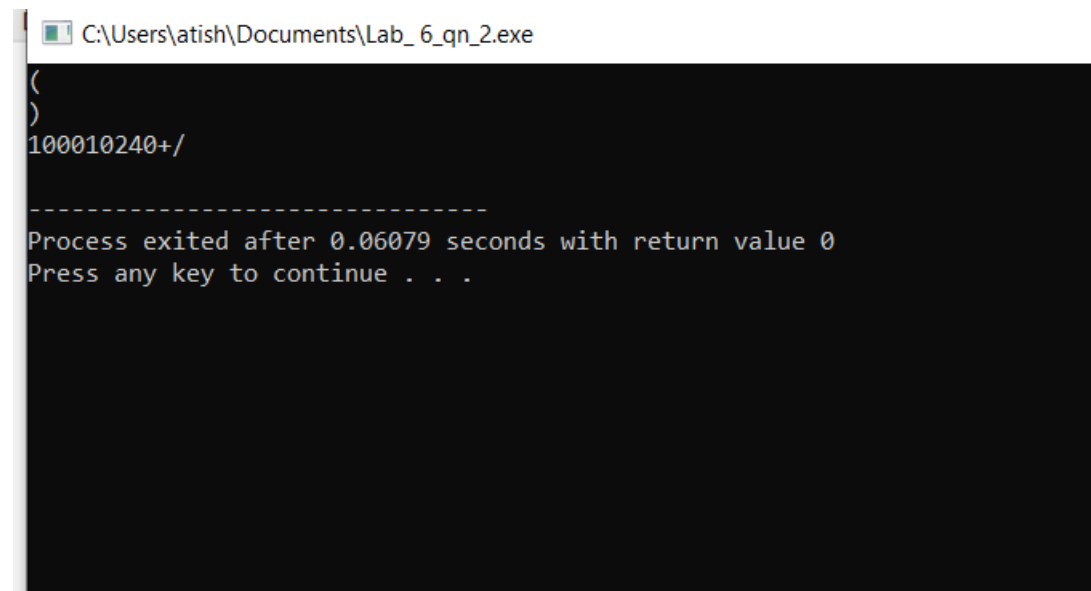
        st.pop();
    }

    st.push(s1);
}

```

```
}  
while (st.empty() != true)  
{  
    postfix = postfix + st.top();  
    st.pop();  
}  
return postfix;  
}  
  
int main()  
{  
    string expression = "1000/(10+240)";  
  
    string ans = infixToPostfix(expression);  
  
    cout<<ans<<endl;  
  
    return 0;  
}
```


Input:-



A screenshot of a Windows command prompt window. The title bar at the top reads "C:\Users\atish\Documents\Lab_6_qn_2.exe". The command prompt shows the following text: an opening parenthesis "(", a closing parenthesis ")", and the input "100010240+/" on separate lines. Below this, a dashed line "-----" is displayed. The next line shows the message "Process exited after 0.06079 seconds with return value 0". The final line is "Press any key to continue . . .".

```
(  
)  
100010240+/  
  
-----  
Process exited after 0.06079 seconds with return value 0  
Press any key to continue . . .
```

Q3. Write a program to implement the following algorithm:

- Start with two indexes one at the left and other at the right end of the array.
- Left index simulate the first stack and second index simulate the right stack.
- If we want to put the element in the first stack then put the element at the left index. Similarly, if we want to put the element in the second stack then put the element at the right index.
- First stack grow towards left and second stack grow towards left. A simple demonstration of dual stack can be observed from the following figure:

Program:-

```
#include<iostream>
```

```
using namespace std;
```

```
class Stack{
```

```
    int* arr;
```

```
    int size;
```

```
    int top1;
```

```
    int top2;
```

```
    public:
```

```
        Stack(int n){
```

```
            size = n;
```

```
            arr = new int[n];
```

```
            top1 = -1;
```

```
            top2 = n;
```

```
        }
```

```
        void push1(int x){
```

```
            if(top1>=top2){
```

```
                cout<<"Stack Overflow"<<endl;
```

```
            }
```

```
            else{
```

```
        top1++;
        arr[top1] = x;
    }
}

void push2(int x){
    if(top1>=top2){
        cout<<"Stack Overflow"<<endl;
    }
    else{
        top2--;
        arr[top2] = x;
    }
}

int pop1(){
    if(top1<0){
        cout<<"Stack Underflow"<<endl;
    }
    else{
        int k = arr[top1];
        top1--;
        return k;
    }
    exit(1);
}

int pop2(){
    if(top2>size-1){
```

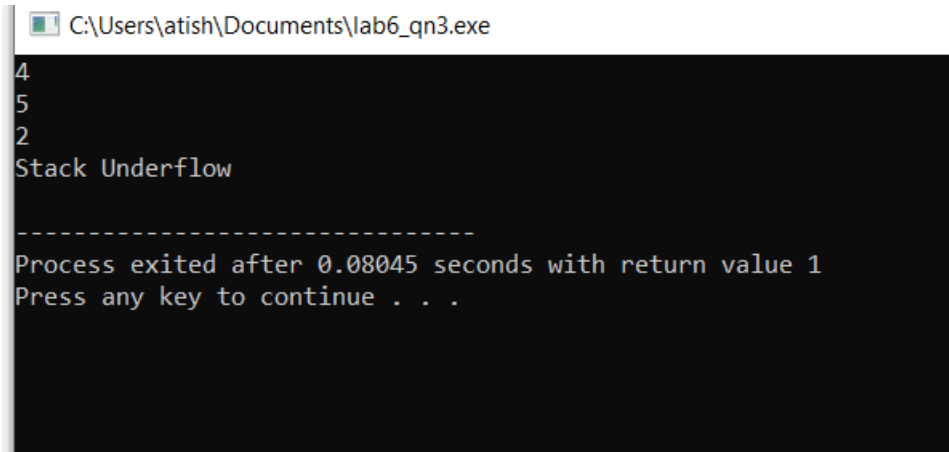
```
        cout<<"Stack Underflow"<<endl;
    }
    else{
        int k = arr[top2];
        top2++;
        return k;
    }
    exit(1);
}

};

int main(){
    Stack st(5);
    st.push1(1);
    st.push2(2);
    st.push1(3);
    st.push1(4);
    st.push2(5);
    cout<<st.pop1()<<endl;
    cout<<st.pop2()<<endl;
    cout<<st.pop2()<<endl;
    cout<<st.pop2()<<endl;

    return 0;
}
```

Input:-



```
C:\Users\atish\Documents\lab6_qn3.exe
4
5
2
Stack Underflow

-----
Process exited after 0.08045 seconds with return value 1
Press any key to continue . . .
```