Bharatiya Vidya Bhavan's
SARDAR PATEL INSTITUTE OF TECHNOLOGY
(Autonomous Institute Affiliated to University of Mumbai)
Munshi Nagar, Andheri (W), Mumbai – 400 058.
Department of AI-ML

Experiment	1
Aim	Implement the given problem statement
Objective	Given a postfix expression as input , evaluate the postfix expression using a stack. Sample input: 231*+9- Infix form: 2 + 3 * 1 - 9 Output: -4
Name	Balla Mahadev Shrikrishna
UCID	2023300010
Class	A
Batch	A
Date of Submission	22-08-24

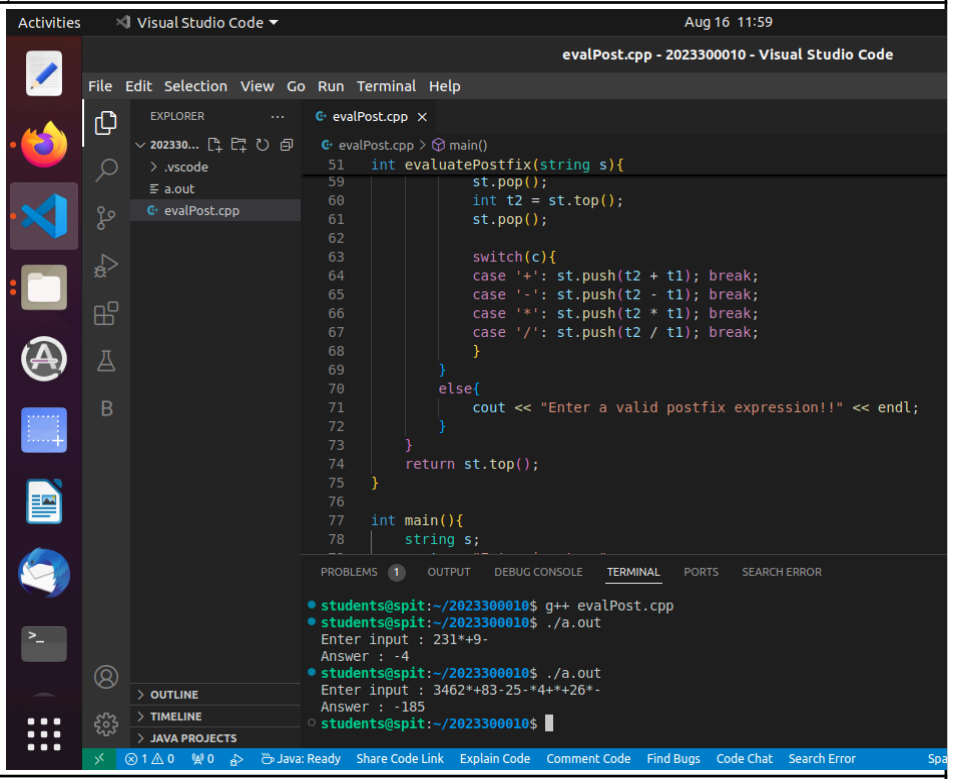
Explanation of the technique used	The evaluatePostfix function iterates through the string(postfix expression) that is passed as an argument while calling. It checks whether the character is a number or not. If so, it pushes that number into the stack. If the character is an operator, it uses two variables to store the values of the top two elements of the stack and pushes the resultant value after carrying out the respective operations. After the execution of this above logic the value of the input postfix expression is stored on the top of the stack.
Program(Code)	

```
#include <bits/stdc++.h>
using namespace std;

class Stackk{
        int topp;
        int arr[30];

public:
        Stackk(){
        topp = -1;
        }

        void push(int n){
        if (topp >= 29){
        cout << "Overflow\n";
        return;
        }
        topp++;
        arr[topp] = n;
        }

        void pop(){
        if(topp < 0){
        cout << "Underflow\n";
```

```
        return;
        }
        topp--;
        }

        int top(){
        if(topp < 0){
        cout << "Underflow\n";
        return -1;
        }
        return arr[topp];
        }

        int size(){
        return topp + 1;
        }

        bool isEmpty(){
        return topp == -1;
        }
};

bool isOperator(char c){
        return c=='*' || c=='/' || c=='+' || c=='-';
}

int evaluatePostfix(string s){
        Stackk st;
        for(auto c : s){
        if(isdigit(c)){
        st.push(c - '0');
        }
        else if(isOperator(c)){
        int t1 = st.top();
        st.pop();
        int t2 = st.top();
        st.pop();

        switch(c){
        case '+': st.push(t2 + t1); break;
        case '-': st.push(t2 - t1); break;
        case '*': st.push(t2 * t1); break;
        case '/': st.push(t2 / t1); break;
        }
        }
        else{
        cout << "Enter a valid postfix expression!!" << endl;
        }
        }
        return st.top();
```

```
}

int main(){
        string s;
        cout << "Enter input : ";
        cin >> s;
        int ans = evaluatePostfix(s);
        cout << "Answer : " << ans << endl;
        return 0;
}
```

Output



Conclusion

The implementation provides a clear method for evaluating postfix expressions, demonstrating the effective use of stack data structures in solving problems.