PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech	**Acad. Yr. 2025-26**	**Semester : I**
Name of the student: Om Jitendra Khalane		**PRN :** 124B1B040
Department: Computer Engineering		**Division :** A

Course Name : Data Structures and Laboratory

Course Code: BCE23PC02

Completion Date : 12/08/2025

Problem Statement:

Write a program for Mathematical Expression Evaluation in Calculator: Implement a calculator that supports evaluation of complex arithmetic expressions using stacks for operands and operators.

Source Code :

https://github.com/omkhalane/DSAL-SY-PCCOE/blob/main/lab_assignments/assignment06.cpp

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

int main()
{
    string exp;
    stack<char> st;
    cout << "Enter Exp: ";
    getline(cin, exp);
```

```
        // getchar();
        for (int i = 0; i < exp.size(); i++)
        {
            if (exp[i] == '+' || exp[i] == '-' || exp[i] == '*' || exp[i] ==
'/')
            {
                int t1 = st.top() - '0';
                st.pop();
                int t2 = st.top() - '0';
                st.pop();
                int sol = 0;
                switch (exp[i])
                {
                case '+':
                    sol = t1 + t2;
                    break;
                case '-':
                    sol = t2 - t1;
                    break;
                case '*':
                    sol = t1 * t2;
                    break;
                case '/':
                    sol = t2 / t1;
                    break;
                default:
                    break;
                }
                st.push(char(sol) + '0');
            }
            else
            {
                st.push(exp[i]);
            }
        }

        cout << "Ans is: " << st.top();

        return 0;
    }
```

Conclusion:

Conclusion for Mathematical Expression Evaluation (Postfix Evaluation):

- Postfix expression evaluation uses **stack-based computation** to process arithmetic expressions efficiently without the need for parentheses.

- Operands are pushed into the stack, and when an operator is encountered, the top two operands are popped, the operation is performed, and the result is pushed back into the stack.

- This continues until the end of the expression, and the final value remaining in the stack is the evaluated result.

Time Complexity (TC):

- **Best / Average / Worst Case:** O(n)
    - Each element (operand/operator) in the expression is processed once, leading to linear time complexity.

Space Complexity (SC):

- **O(n)** due to the use of a stack for storing operands during evaluation.

- The space depends on the number of operands in the expression.