**Experiment 4**

**AIM:** Write a program to evaluate a postfix expression using stack.

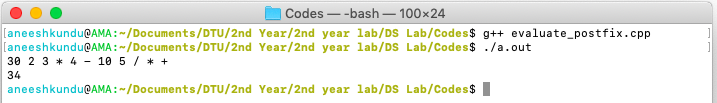
**THEORY:** The Postfix notation is used to represent algebraic expressions. The expressions written in postfix form are evaluated faster compared to infix notation as parenthesis are not required in postfix.

Following is algorithm for evaluation postfix expressions.

1. Create a stack to store operands (or values).
2. Scan the given expression and do following for every scanned element.
   1. If the element is a number, push it into the stack
   2. If the element is a operator, pop operands for the operator from stack. Evaluate the operator and push the result back to the stack.
3. When the expression is ended, the number in the stack is the final answer

**Code:**

1. #include < iostream >
2. #include < vector >
3. #include < stack >
4. #include < string >
5. #include < sstream >
6. #include < cctype >
7. using namespace std;
8. vector < string > split(string pf) {
9. istringstream iss(pf);
10. string token;
11. vector < string > tokens;
12. while (getline(iss, token, ' ')) tokens.push\_back(token);
13. return tokens;
14. }
15. int perform\_op(int a, int b, string op) {
16. if (op == "/") return a / b;
17. else if (op == "\*") return a \* b;
18. else if (op == "+") return a + b;
19. else return a - b;
20. }
21. int evaluate\_postfix(string pf) {
22. vector < string > postfix = split(pf);
23. stack < int > s;
24. int b, a;
25. for (int i = 0; i < postfix.size(); i++) {
26. if (isdigit(postfix[i][0])) s.push(stoi(postfix[i]));
27. else {
28. b = s.top();
29. s.pop();
30. a = s.top();
31. s.pop();
32. s.push(perform\_op(a, b, postfix[i]));
33. }
34. }
35. return s.top();
36. }
37. int main() {
38. string postfix;
39. getline(cin, postfix);
40. cout << evaluate\_postfix(postfix) << endl;
41. return 0;
42. }

**Output:**