#include <iostream>

using namespace std;

template <typename T>

class Stack

{

    int top = -1, size;

    T \*arr;

public:

    Stack(int a)

    {

        size = a;

        arr = new T[a];

    }

    bool isEmpty()

    {

        return top == -1;

    }

    void push(T element)

    {

        arr[++top] = element;

    }

    T pop()

    {

        return arr[top--];

    }

    T peek()

    {

        return arr[top];

    }

};

bool isOperator(char c)

{

    return (c == '+' || c == '-' || c == '/' || c == '\*');

}

bool checkPrecedence(char a, char b)

{

    if (a == '\*' && isOperator(b))

        return true;

    else if (a == '/' && isOperator(b))

        return true;

    else if (a == '+' && (b == '+' || b == '-'))

        return true;

    else if (a == '-' && (b == '+' || b == '-'))

        return true;

    return false;

}

string infixToPostfix(string infix)

{

    int n = infix.length();

    Stack<char> operator\_stack(n);

    string postfix = "";

    for (int i = 0; i < n; i++)

    {

        if (!isOperator(infix[i]))

            postfix += infix[i];

        else

        {

            while (!operator\_stack.isEmpty() && checkPrecedence(operator\_stack.peek(), infix[i]))

                postfix += operator\_stack.pop();

            operator\_stack.push(infix[i]);

        }

    }

    while (!operator\_stack.isEmpty())

        postfix += operator\_stack.pop();

    return postfix;

}

int eval(int first, int second, char op)

{

    switch (op)

    {

    case '/':

        return first / second;

    case '\*':

        return first \* second;

    case '+':

        return first + second;

    case '-':

        return first - second;

    default:

        return 0;

    }

}

int evaluatePostfix(string postfix)

{

    int n = postfix.length();

    Stack<int> operand\_stack(n);

    int first, second;

    for (int i = 0; i < n; i++)

    {

*// operand*

        if (!isOperator(postfix[i]))

            operand\_stack.push(postfix[i] - '0'); *// convert to ints*

        else

        {

            second = operand\_stack.pop();

            first = operand\_stack.pop();

            operand\_stack.push(eval(first, second, postfix[i])); *// push the result of first op second*

        }

    }

    return operand\_stack.pop();

}

int main()

{

    string infix, postfix;

    cout << "Enter infix : " << endl;

    cin >> infix;

    postfix =  infixToPostfix(infix);

    cout << "Postfix : " << postfix << endl;

    cout << "Value: " << evaluatePostfix(postfix) << endl;

    return 0;

}

