#include<iostream>

#include<cctype>

#include<stack>

using namespace std;

// returns the value when a specific operator

// operates on two operands

int eval(int op1, int op2, char operate) {

switch (operate) {

case '\*': return op2 \* op1;

case '/': return op2 / op1;

case '+': return op2 + op1;

case '-': return op2 - op1;

default : return 0;

}

}

int getWeight(char ch) {

switch (ch) {

case '/':

case '\*': return 2;

case '+':

case '-': return 1;

default : return 0;

}

}

// evaluates the postfix operation

// this module neither supports multiple digit integers

// nor looks for valid expression

// However it can be easily modified and some additional

// code can be added to overcome the above mentioned limitations

// it's a simple function which implements the basic logic to

// evaluate postfix operations using stack

int evalPostfix(char postfix[], int size) {

stack<int> s;

int i = 0;

char ch;

int val;

while (i < size) {

ch = postfix[i];

if (isdigit(ch)) {

// we saw an operand

// push the digit onto stack

s.push(ch-'0');

}

else {

// we saw an operator

// pop off the top two operands from the

// stack and evalute them using the current

// operator

int op1 = s.top();

s.pop();

int op2 = s.top();

s.pop();

val = eval(op1, op2, ch);

// push the value obtained after evaluating

// onto the stack

s.push(val);

}

i++;

}

return val;

}

// main

int main() {

char postfix[] = {'a','b','c','+','\*','d','/'};

int i=0;

// cout<<"Postfix 0"<<postfix[i];

while(postfix[i]!='\0')

{

if(getWeight(postfix[i])==0)

{

cout<<" Enter value for "<<postfix[i]<<" = ";

cin>>postfix[i];

}

i++;

}

int size = sizeof(postfix);

int val = evalPostfix(postfix, size);

cout<<"\nExpression evaluates to "<<val;

cout<<endl;

return 0;

}

Output:

Enter value for a = 2

Enter value for b = 3

Enter value for c = 4

Enter value for d = 5

Enter value for � = 2

Enter value for  = 4

Expression evaluates to 2