#include <iostream>

#include <string>

#include <cstdlib>

#include <cmath>

#include <stack>

#include <cctype>

using namespace std;

int calc(char token, int num1, int num2);

int getweight(char op);

bool isOperator(char ch){

if( ch == '+' ||ch == '-' ||ch == '\*'||ch == '/'||ch == '^')

return true;

return false;

}

bool hasPrec(char op1, char op2){

int op\_pr1 = getweight(op1);

int op\_pr2 = getweight(op2);

if(op\_pr1 == op\_pr2){

if(op1 =='^') return false;

return true;

}

return op\_pr1 >= op\_pr2 ? true:false;

}

string infix2postfix(string infix){

stack<char> tempStk;

string opStk, postfixQ;

for( int i=0; i<=infix.size()-1;i++){

if(isalnum(infix[i])){

postfixQ += infix[i];

}

if(isOperator(infix[i])){

while(!tempStk.empty()&&hasPrec(tempStk.top(), infix[i])){

postfixQ += tempStk.top();

opStk.erase(opStk.size()-1);

tempStk.pop();

cout << " \n op stack: " << opStk;

cout << "\t\n postfix queue :" << postfixQ;

}

tempStk.push(infix[i]);

opStk +=infix[i];

cout << " \n op stack: " << opStk;

cout << "\t\n postfix queue :" << postfixQ;

}

}

while(!tempStk.empty()){

postfixQ += tempStk.top();

tempStk.pop();

cout << " \n op stack: " << opStk;

cout << "\t\n postfix queue :" << postfixQ;

}

return postfixQ;

}

int getweight(char op){

int weight = -1;

switch(op){

case '+' :weight = 1;break;

case '-' :weight = 1; break;

case '\*' :weight = 2; break;

case '/' :weight = 2; break;

case '^' :weight = 3; break;

}

return weight;

}

int main() {

string postfix;

string infix;

while(true) {

cout << "\nEnter a postfix expression, q to quit: ";

getline(cin, infix);

if(infix == "q") break;

//cout << "\ninfix => " << infix << endl;

postfix= infix2postfix(infix);

cout << "\n infix => " << infix << "\n postfix => " << postfix <<endl;

// calculate Postfix

deque<char> postfixStk;

for(auto item:postfix) postfixStk.push\_back(item);

stack<int> stk; // operands

int num1, num2, result = 0, step = 1;

while( !postfixStk.empty() ) {

char token = postfixStk.front();

if(isalnum(token)) {

stk.push(token-'0');

postfixStk.pop\_front();

}

else if(token=='+' || token== '-' ||

token=='\*' || token== '/' || token=='^') {

postfixStk.pop\_front();

num2 = stk.top(); stk.pop(); // right operand on top of the left operand

num1 = stk.top(); stk.pop();

result = calc(token, num1, num2);

stk.push(result);

cout << "Step " << step << ": " << result

<< " = " << num1 << token << num2 << endl;

step++;

}

}cout <<"answer: "<< result;

}

}

int calc(char token, int num1, int num2){

int result;

switch(token) {

case '\*': result = num1 \* num2; break;

case '/': result = num1 / num2; break;

case '+': result = num1 + num2; break;

case '-': result = num1 - num2; break;

case '^': result = pow(num1, num2); break;

}

return result;

}

