#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include <assert.h>

struct token{

int type;//1变量2常数3运算符

int th;//字符串长度

int xsdwz;//小数点位置

char str[40];//名称

};

struct token tokens[520];

struct assign{

char name[32];

int val;

double valflo;

int type;//0;1

};

struct assign assignments[130];

int yf = 1;//语法分析

int cf = 1;//词法分析

int number = 0;//变量个数

int fds = 0;//判断是否出现浮点数

int check\_kuohuo(int l,int r);

int eval(int l,int r);

double evalflo(int l,int r);

int find(int l,int r);

int main(void)

{

char \*p;

p = (char \*) malloc(1000 \* sizeof(char));

int n = 0;

while(scanf("%s",p) != EOF)

{

char ch = getchar();

int xsd = 0;

int pd = 0;

//词法分析

int len = strlen(p);

if(len == 1)

{

pd = 1;

if(\*p == '.')

cf = 0;

else if(\*p == '+' || \*p == '-' || \*p == '\*' || \*p == '/' || \*p == '(' || \*p == ')' || \*p == '=')

{

tokens[n].type = 3;

tokens[n].th = len;

tokens[n].str[0] = \*p;

tokens[n].str[1] = '\0';

n++;

}

else if(isdigit(\*p) != 0)

{

tokens[n].type = 2;

tokens[n].th = len;

tokens[n].str[0] = \*p;

tokens[n].str[1] = '\0';

n++;

}

else

{

tokens[n].type = 1;

tokens[n].th = len;

tokens[n].str[0] = \*p;

tokens[n].str[1] = '\0';

for(int check = 0;check<130;check++)

{

if(strcmp(tokens[n].str,assignments[check].name) == 0)

{

if(assignments[check].type == 1)

fds++;

break;

}

}

n++;

}

}

else

{

if(isalpha(\*p) != 0 || \*p == '\_')

{

pd = 1;

for(int i = 0;i<len;i++)

{

if(isalnum(\*(p+i)) == 0 && \*(p+i) != '\_')

{

pd = 0;

break;

}

}

if(pd == 1)

{

tokens[n].type = 1;

tokens[n].th = len;

for(int i = 0;i<len;i++)

tokens[n].str[i] = \*(p+i);

tokens[n].str[len] = '\0';

for(int check = 0;check<130;check++)

{

if(strcmp(tokens[n].str,assignments[check].name) == 0)

{

if(assignments[check].type == 1)

fds++;

}

}

n++;

}

}

if(isdigit(\*p) != 0)

{

if(\*p != '0'||(\*p=='0'&&\*(p+1)== '.')){

pd = 1;

for(int i = 0;i<len;i++)

{

if(isdigit(\*(p+i)) == 0)

{

if(\*(p+i) == '.')

xsd++;

else{

pd = 0;

break;

}

}

}

if(xsd>1 || \*(p+len-1) == '.')

pd = 0;

if(pd == 1)

{

tokens[n].type = 2;

tokens[n].th = len;

for(int i = 0;i<len;i++)

{

tokens[n].str[i] = \*(p+i);

if(\*(p+i) == '.')

{

tokens[n].xsdwz=i;

fds++;

}

}

//if(xsd == 1 && (len-tokens[n].xsdwz-1)<6)

// {

// for(int tap = len;tap<=tokens[n].xsdwz+6;tap++)

// tokens[n].str[tap] = '0';

// tokens[n].str[tokens[n].xsdwz+7] = '\0';

// }

//else{

tokens[n].str[len] = '\0';

//}

n++;

}

}

}

if(pd == 0)

cf = 0;

}

//词法分析 cf=0词法错误 =1词法正确

if(ch == '\n')//刚好读入一行

{

if(cf == 0)

{

printf("Error\n");

cf = 1;

n = 0;

}

else

{

if(fds == 0)

{//

int dan = eval(0,n-1);

if(yf == 0)

{

printf("Error\n");

n = 0;

yf = 1;

}

else

{

printf("%d\n",dan);

n = 0;

}

}//

else

{//

double dan\_f = evalflo(0,n-1);

fds = 0;

if(yf == 0)

{

printf("Error\n");

n = 0;

yf = 1;

}

else

{

if(dan\_f == (int)(1 \* dan\_f))

printf("%d\n",(int)dan\_f);

else

printf("%lf\n",dan\_f);

n = 0;

}

}//

}

}

}

return 0;

}

//处理括号

int check\_kuohuo(int l,int r)

{

if(tokens[l].str[0] != '(' || tokens[r].str[0] != ')')

return 0;

else

{

int khjc = 0;

for(int i = l+1;i<r;i++)

{

if(tokens[i].str[0] == '(')

khjc++;

else if(tokens[i].str[0] == ')')

khjc--;

if(khjc < 0)

return 0;

}

if(khjc != 0)

return 0;

}

return 1;

}

//整数计算

int eval(int l,int r)

{

if(l>r)

{

yf = 0;

return 0;

}

else if(l == r)

{

if(tokens[l].type == 2)

{

int num = tokens[l].th;

int total = 0;

int x = num;

int y = 1;

for(int i = 0;i<num;i++)

{

for(int t = 1;t<x;t++)

y = 10 \* y;

total = total + y \* (tokens[l].str[i] - 48);

x--;

y = 1;

}

return total;

}

else if(tokens[l].type == 1)

{

for(int i = 0;i<number;i++)

{

if(strcmp(tokens[l].str,assignments[i].name) == 0)

return assignments[i].val;

}

yf = 0;

return 0;

}

else

{

yf = 0;

return 0;

}

}

else if(check\_kuohuo(l,r) == 1)

return eval(l+1,r-1);

else

{

int zysf = find(l,r);

if(zysf == -1)

{

yf = 0;

return 0;

}

switch(tokens[zysf].str[0])

{

case '+': return eval(l,zysf-1) + eval(zysf+1,r);

case '-':

if(zysf == l)

return (-1) \* eval(zysf+1,r);

else

return eval(l,zysf-1) - eval(zysf+1,r);

case '\*': return eval(l,zysf-1) \* eval(zysf+1,r);

case '/': return eval(l,zysf-1) / eval(zysf+1,r);

case '=': for(int test=0;test<zysf;test++)

{

if(tokens[test].str[0]== '(' ||tokens[test].str[0]== ')' ||tokens[test].str[0]== '+' ||tokens[test].str[0]== '-' ||tokens[test].str[0]== '\*' ||tokens[test].str[0]== '/'

|| (tokens[test].str[0]>=48&&tokens[test].str[0]<=57))

{

yf = 0;

return 0;

}

}

int pdtm = 0;

int tmwz;

for(int tm = 0;tm<number;tm++)

{

if(strcmp(tokens[zysf-1].str,assignments[tm].name) == 0)

{

pdtm = 1;

tmwz = tm;

break;

}

}

if(pdtm == 1)

{

assignments[tmwz].val = eval(zysf+1,r);

assignments[tmwz].type = 0;

}

else{

int cd = tokens[zysf-1].th;

for(int i = 0;i<cd;i++)

assignments[number].name[i] = tokens[zysf-1].str[i];

assignments[number].val = eval(zysf+1,r);

assignments[number].type = 0;

number++;

}

return eval(l,zysf-1);

default:assert(0);

}

}

}

//浮点数计算

double evalflo(int l,int r)

{

if(l>r)

{

yf = 0;

return 0;

}

else if(l == r)

{

if(tokens[l].type == 2)

{

int csfds = 0;

for(int i = 0;i<tokens[l].th;i++)

{

if(tokens[l].str[i] == '.')

{

csfds = 1;

break;

}

}

if(csfds == 1)

{

int num = tokens[l].xsdwz;

double total = 0.000000;

int x = num;

double y = 1.000000;

for(int i = 0;i<num;i++)

{

for(int t = 1;t<x;t++)

y = 10.0 \* y;

total = total + y \* (tokens[l].str[i] - 48);

x--;

y = 1.000000;

}

double kk = 0.1;

for(int i = 1;tokens[l].str[num+i] != '\0';i++)

{

total = total + kk \* (tokens[l].str[num+i] - 48);

kk = kk / 10.00;

}

return (double)total;

}

else

{

double tot = 0.000000;

int z = tokens[l].th;

int zz = z;

double u = 1.000000;

for(int i = 0;i<z;i++)

{

for(int t = 1;t<zz;t++)

u = 10.0 \* u;

tot = tot + u \* (tokens[l].str[i] - 48);

zz--;

u = 1.000000;

}

return (double)tot;

}

}

else if(tokens[l].type == 1)

{

for(int i = 0;i<number;i++)

{

if(strcmp(tokens[l].str,assignments[i].name) == 0){

if(assignments[i].type == 1)

return (double)assignments[i].valflo;

else

return (double)(1.0 \* assignments[i].val);

}

}

yf = 0;

return 0;

}

else

{

yf = 0;

return 0;

}

}

else if(check\_kuohuo(l,r) == 1)

return evalflo(l+1,r-1);

else

{

int zysf = find(l,r);

if(zysf == -1)

{

yf = 0;

return 0;

}

switch(tokens[zysf].str[0])

{

case '+': return evalflo(l,zysf-1) + evalflo(zysf+1,r);

case '-':

if(zysf == l)

return (-1.000000) \* evalflo(zysf+1,r);

else

return evalflo(l,zysf-1) - evalflo(zysf+1,r);

case '\*': return evalflo(l,zysf-1) \* evalflo(zysf+1,r);

case '/':

if(l == zysf-1 && r == zysf+1)

{

if(strchr(tokens[l].str,'.')==NULL&&strchr(tokens[r].str,'.')==NULL)

return (double)((int)evalflo(l,zysf-1)/(int)evalflo(zysf+1,r));

}

return evalflo(l,zysf-1) / evalflo(zysf+1,r);

case '=': for(int test=0;test<zysf;test++)

{

if(tokens[test].str[0]== '(' ||tokens[test].str[0]== ')' ||tokens[test].str[0]== '+' ||tokens[test].str[0]== '-' ||tokens[test].str[0]== '\*' ||tokens[test].str[0]== '/'

|| (tokens[test].str[0]>=48&&tokens[test].str[0]<=57))

{

yf = 0;

return 0;

}

}

int pdtm = 0;

int tmwz;

for(int tm = 0;tm<number;tm++)

{

if(strcmp(tokens[zysf-1].str,assignments[tm].name) == 0)

{

pdtm = 1;

tmwz = tm;

break;

}

}

if(pdtm == 1)

{

assignments[tmwz].valflo = evalflo(zysf+1,r);

assignments[tmwz].type = 1;

}

else{

int cd = tokens[zysf-1].th;

for(int i = 0;i<cd;i++)

assignments[number].name[i] = tokens[zysf-1].str[i];

assignments[number].valflo = evalflo(zysf+1,r);

assignments[number].type = 1;

number++;

}

return evalflo(l,zysf-1);

default:assert(0);

}

}

}

int find(int l,int r)

{

int tap = -1;

int Tap = -1;

int TAP = -1;

int jl = 0;

int dea = r;

for(int i = l;i<=r;i++)

{

if(tokens[i].str[0] == '+')

tap = i;

else if(tokens[i].str[0] == '-')

{

if(i == l)

tap = i;

else

{

if(tokens[i-1].str[0] != '+'&&tokens[i-1].str[0] != '-'&&tokens[i-1].str[0] != '\*'&&tokens[i-1].str[0] != '/')

tap = i;

}

}

else if(tokens[i].str[0] == '\*' || tokens[i].str[0] == '/')

Tap = i;

else if(tokens[i].str[0] == '=')

TAP = i;

if(tokens[i].str[0] == '(')

jl++;

while(jl>0 && dea > 0)

{

i++;

if(tokens[i].str[0] == '(')

jl++;

else if(tokens[i].str[0] == ')')

jl--;

dea--;

}

}

if(TAP != -1)

return TAP;

if(tap != -1)

{

if(tap == l && Tap != -1)

return Tap;

else

return tap;

}

else if(tap == -1 && Tap != -1)

{

return Tap;

}

return -1;

}