#include<iostream>

#include<math.h>

using namespace std;

class Poly;

class PolyNode

{

friend Poly;

private:

double coef;

int exp;

PolyNode \*link;

public:

PolyNode(PolyNode \*temp = NULL){link = temp;};

PolyNode(double coef,int exp,PolyNode \*temp = NULL){this->coef = coef;this->exp = exp;link = temp;}

~PolyNode(){};

};

class Poly

{

private:

PolyNode \*first;

public:

Poly(){first = new PolyNode();};

~Poly(){MakeEmpty();};

/\*清空多项式\*/

void MakeEmpty();

/\*返回多项式长度\*/

int Length();

/\*找到exp的位置并返回PolyNode指针\*/

PolyNode \*Find(int exp);

/\*新项的插入\*/

int Insert(double coef,int exp);

/\*移除特定的某一项\*/

int Remove(double coef,int exp);

/\*判断多项式是否为空\*/

int IsEmpty();

/\*输入多项式\*/

void input(int endTag);

/\*输出多项式\*/

void output();

/\*给多项式输入数值后计算结果\*/

double Count(double x);

/\*原多项式加多项式\*/

int add(Poly ploy);

/\*原多项式减多项式\*/

int reduce(Poly poly);

/\*多项式相乘\*/

int multiply(Poly ploy);

};

void Poly::MakeEmpty()

{

PolyNode \*temp;

while(first->link!=NULL)

{

temp = first->link;

first->link = temp->link;

delete temp;

}

}

int Poly::Length()

{

int count = 0;

PolyNode \*temp = first->link;

while (temp!=NULL)

{

count++;

temp = temp->link;

}

return count;

}

PolyNode \*Poly::Find(int exp)

{

if(exp<0)return 0;

PolyNode \*temp = first;

while(temp->link!=0)

{

if(temp->link->exp==exp)

{

return temp->link;

}

else

{

temp = temp->link;

}

}

return 0;

}

int Poly::Insert(double coef, int exp)

{

if (coef == 0) {

return 1;

}

PolyNode \*prev = first;

PolyNode \*current = first->link;

while (current != NULL && current->exp < exp) {//判断当输入的exp小于等于原多项式中的某一项时就可以插入这一位置

prev = current;

current = current->link;

}

if (current != NULL && current->exp == exp) {//如果exp相同则合并系数即可

current->coef += coef;

}

else {

PolyNode \*new\_node = new PolyNode(coef, exp);//如果不相同则新建插入

new\_node->link = current;

prev->link = new\_node;

}

return 1;

}

int Poly::Remove(double coef,int exp)

{

PolyNode \*temp = Find(exp);//查询exp

if(temp!=NULL)//如果存在则执行删除

{

if(temp->coef==coef)//系数相同则直接删除

{

PolyNode \*prev = first;

PolyNode \*current = first->link;//两个结点指针一前一后遍历

while (current->coef!=coef)

{

prev = current;

current = current->link;

}

prev->link = current->link;

delete current;

}

else if(temp->coef!=coef)//系数不同则执行相减

{

temp->coef = temp->coef-coef;

}

return 1;

}

return 0;

}

int Poly::IsEmpty()

{

return(first->link == NULL?1:0);

}

void Poly::input(int endTag)

{

double coef = 0.0;

int exp = 0;

do {

cin>>coef>>exp;

if (coef != endTag) {

Insert(coef, exp);

}

} while (coef != endTag);

}

void Poly::output()

{

PolyNode \*temp = first->link;

while(temp!=NULL)

{

cout<<temp->coef<<"\*x^"<<temp->exp<<endl;

temp = temp->link;

}

}

double Poly::Count(double x)

{

PolyNode \*temp = first;

double result = 0;

while (temp->link!=NULL)

{

temp = temp->link;

result += temp->coef \* pow(x, temp->exp);

}

return result;

}

int Poly::add(Poly poly)

{

if(poly.Length()!=0)

{

PolyNode \*current = poly.first->link;

while (current != NULL)

{

Insert(current->coef,current->exp);

current = current->link;

}

return 1;

}

return 0;

}

int Poly::reduce(Poly poly)

{

if(poly.Length()!=0)

{

PolyNode \*current = poly.first->link;

while (current != NULL)

{

Remove(current->coef,current->exp);

current = current->link;

}

return 1;

}

return 0;

}

int Poly::multiply(Poly poly)

{

if(poly.Length()!=0)

{

Poly result;

PolyNode \*current = poly.first->link;

while (current != NULL)

{

for (PolyNode \*node = first->link; node != NULL; node = node->link)

{

double coef = node->coef \* current->coef;

int exp = node->exp + current->exp;

result.Insert(coef, exp);

}

current = current->link;

}

MakeEmpty();

PolyNode \*temp = result.first->link;

while (temp != nullptr)

{

Insert(temp->coef, temp->exp);

temp = temp->link;

}

return 1;

}

return 0;

}

int main()

{

Poly myPoly;

Poly myPoly2;

myPoly.input(100);

myPoly.output();

cout<<"--------"<<endl;

myPoly2.input(100);

myPoly2.output();

//cout<<"--------"<<endl;

//myPoly.reduce(myPoly2);

//myPoly.output();

cout<<"--------"<<endl;

myPoly.multiply(myPoly2);

myPoly.output();

/\*myPoly.output();

cout<<"--------"<<endl;

myPoly.Insert(3,3);

myPoly.output();

cout<<"--------"<<endl;

myPoly.Remove(3,3);

myPoly.output();

cout<<"--------"<<endl;

cout<<myPoly.Count(1)<<endl;

cout<<"--------"<<endl;\*/

return 1;

}