#include<stdio.h>

#include<stdlib.h>

typedef struct Node \*PtrToNode;

struct Node{

int c;

int e;

PtrToNode Next;

};

typedef PtrToNode List;

List ReadPoly();

List Add(List P1,List P2);

void Attach(int c,int e,List\*pRear);

List Mult(List P1,List P2);

void PrintPoly(List P);

int main()

{

List P1,P2,PP,PS;

P1 = ReadPoly();

P2 = ReadPoly();

PP = Mult(P1,P2);

PrintPoly(PP);

PS = Add(P1,P2);

PrintPoly(PS);

return 0;

}

List ReadPoly()

{

List P,Rear,t;

int c,e,N;

scanf("%d",&N);

P = (List)malloc(sizeof(struct Node));

P->Next = NULL;

Rear = P;

while(N--)

{

scanf("%d %d",&c,&e);

if(c==0)

e=0;

Attach(c,e,&Rear);

}

t =P;

P = P->Next;

free(t);

return P;

}

void Attach(int c,int e,List\*pRear)

{

List P;

P = (List)malloc(sizeof(struct Node));

P->c = c;

P->e = e;

P->Next = NULL;

(\*pRear)->Next = P;

\*pRear = P;

}

List Add(List P1,List P2)

{

List P,rear,temp,t1,t2;

int sum;

t1 = P1;

t2 = P2;

P = (List)malloc(sizeof(struct Node));

P->Next = NULL;

rear = P;

while(t1 && t2)

{

if(t1->e == t2->e)

{

sum = t1->c + t2->c;

if(sum)

Attach(sum,t1->e,&rear);

t1 = t1->Next;

t2 = t2->Next;

}

else if(t1->e > t2->e)

{

Attach(t1->c,t1->e,&rear);

t1 = t1->Next;

}

else

{

Attach(t2->c,t2->e,&rear);

t2 = t2->Next;

}

}

for(;t1;t1=t1->Next)

Attach(t1->c,t1->e,&rear);

for(;t2;t1=t2->Next)

Attach(t2->c,t2->e,&rear);

rear->Next = NULL;

temp = P;

P = P->Next;

free(temp);

return P;

}

List Mult(List P1,List P2)

{

List P,Rear,t1,t2,t;

int x,y;

if(!P1 || !P2)

return NULL;

P = (List)malloc(sizeof(struct Node));

P->Next = NULL;

Rear = P;

t1 = P1;

t2 = P2;

while(t2)

{

x=t1->c \* t2->c;

y=t1->e + t2->e;

Attach(x,y,&Rear);

t2=t2->Next;

}

t1 = t1->Next;

while(t1)

{

t2 = P2;

Rear = P;

while(t2)

{

x=t1->c \* t2->c;

y=t1->e + t2->e;

while((Rear->Next) && (Rear->Next->e > y))

Rear = Rear->Next;

if((Rear->Next) && (Rear->Next->e == y))

{

if((Rear->Next->c + x) != 0)

Rear->Next->c += x;

else

{

t = Rear->Next;

Rear->Next = t->Next;

free(t);

}

}

else

{

t = (List)malloc(sizeof(struct Node));

                t->c = x;

                t->e = y;

                t->Next = Rear->Next;

                Rear->Next = t;

                Rear = Rear->Next;

}

t2=t2->Next;

}

t1 = t1->Next;

}

t = P;

P=P->Next;

free(t);

return P;

}

void PrintPoly(List P)

{

int flag=0;

if(!P)

{

printf("0 0\n");

return;

}

while(P)

{

if(!flag)

flag= 1;

else

printf(" ");

printf("%d %d",P->c,P->e);

P=P->Next;

}

printf("\n");

}