Educational Codeforces Round 108 (Rated for Div. 2)

A. Red and Blue Beans

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have r red and b blue beans. You'd like to distribute them among several (maybe, one) packets in such a way that each packet:

- has at least one red bean (or the number of red beans $r_i \geq 1$);
- has at least one blue bean (or the number of blue beans $b_i \geq 1$);
- the number of red and blue beans should differ in no more than d (or $|r_i - b_i| \leq d$)

Can you distribute all beans?

Input

The first line contains the single integer t ($1 \leq t \leq 1000$) — the number of test cases.

The first and only line of each test case contains three integers r, b, and d ($1 \leq r, b \leq 10^9$; $0 \leq d \leq 10^9$) — the number of red and blue beans and the maximum absolute difference in each packet.

Output

For each test case, if you can distribute all beans, print YES. Otherwise, print NO.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES are all recognized as positive answer).

Example

input
4 1 1 0 2 7 3 6 1 4 5 4 0
output
YES YES NO NO

Note

In the first test case, you can form one packet with 1 red and 1 blue bean. The absolute difference $|1 - 1| = 0 \leq d$.

In the second test case, you can form two packets: 1 red and 4 blue beans in the first packet and 1 red and 3 blue beans in the second one.

In the third test case, since $b = 1$, you can form only one packet with 6 red and 1 blue beans. The absolute difference $|6 - 1| = 5 > d$.

In the fourth test case, since $d = 0$ so each packet should contain the same number of red and blue beans, but $r \neq b$.

B. The Cake Is a Lie

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There is a $n \times m$ grid. You are standing at cell $(1, 1)$ and your goal is to finish at cell (n, m).

You can move to the neighboring cells to the right or down. In other words, suppose you are standing at cell (x, y). You can:

- move right to the cell $(x, y + 1)$ — it costs x burles;
- move down to the cell $(x + 1, y)$ — it costs y burles.

Can you reach cell (n, m) spending **exactly** k burles?

Input

The first line contains the single integer t ($1 \leq t \leq 100$) — the number of test cases.

The first and only line of each test case contains three integers n, m, and k ($1 \le n, m \le 100$; $0 \le k \le 10^4$) — the sizes of grid and the exact amount of money you need to spend.

Output
For each test case, if you can reach cell (n, m) spending **exactly** k burles, print YES. Otherwise, print NO.

You may print every letter in any case you want (so, for example, the strings yEs, yes, Yes and YES are all recognized as positive answer).

Example

input
6 1 1 0 2 2 2 2 2 3 2 2 4 1 4 3 100 100 10000
output
YES NO YES NO YES NO

Note
In the first test case, you are already in the final cell, so you spend 0 burles.

In the second, third and fourth test cases, there are two paths from $(1, 1)$ to $(2, 2)$: $(1, 1) \to (1, 2) \to (2, 2)$ or $(1, 1) \to (2, 1) \to (2, 2)$. Both costs $1 + 2 = 3$ burles, so it's the only amount of money you can spend.

In the fifth test case, there is the only way from $(1, 1)$ to $(1, 4)$ and it costs $1 + 1 + 1 = 3$ burles.

C. Berland Regional

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is an organizer of a Berland ICPC regional event. There are n universities in Berland numbered from 1 to n. Polycarp knows all competitive programmers in the region. There are n students: the i-th student is enrolled at a university u_i and has a programming skill s_i.

Polycarp has to decide on the rules now. In particular, the number of members in the team.

Polycarp knows that if he chooses the size of the team to be some integer k, each university will send their k strongest (with the highest programming skill s) students in the first team, the next k strongest students in the second team and so on. If there are fewer than k students left, then the team can't be formed. Note that there might be universities that send zero teams.

The strength of the region is the total skill of the members of all present teams. If there are no teams present, then the strength is 0.

Help Polycarp to find the strength of the region for each choice of k from 1 to n.

Input
The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

The first line of each testcase contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of universities and the number of students.

The second line of each testcase contains n integers $u_1, u_2, \ldots, u_n$ ($1 \le u_i \le n$) — the university the i-th student is enrolled at.

The third line of each testcase contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 10^9$) — the programming skill of the i-th student.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$.

Output
For each testcase print n integers: the strength of the region — the total skill of the members of the present teams — for each choice of team size k.

Example

input
4 7 1 2 1 2 1 2 1 6 8 3 1 5 1 5 10

```
1 1 1 2 2 2 2 3 3 3
3435 3014 2241 2233 2893 2102 2286 2175 1961 2567
6
3 3 3 3 3 3
5 9 6 7 9 7
1
1
3083
```

output

```
29 28 26 19 0 0 0
24907 20705 22805 9514 0 0 0 0 0 0
43 43 43 32 38 43
3083
```

Note

In the first testcase the teams from each university for each k are:

- $k = 1$:
  - university 1: $[6], [5], [5], [3]$;
  - university 2: $[8], [1], [1]$;
- $k = 2$:
  - university 1: $[6, 5], [5, 3]$;
  - university 2: $[8, 1]$;
- $k = 3$:
  - university 1: $[6, 5, 5]$;
  - university 2: $[8, 1, 1]$;
- $k = 4$:
  - university 1: $[6, 5, 5, 3]$;

D. Maximum Sum of Products

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two integer arrays a and b of length n.

You can reverse **at most one** subarray (continuous subsegment) of the array a.

Your task is to reverse such a subarray that the sum $\sum_{i=1}^{n} a_i \cdot b_i$ is **maximized**.

Input

The first line contains one integer n ($1 \le n \le 5000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^7$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^7$).

Output

Print single integer — maximum possible sum after reversing **at most one** subarray (continuous subsegment) of a.

Examples

input

```
5
2 3 2 1 3
1 3 2 4 2
```

output

```
29
```

input

```
2
13 37
2 4
```

output

```
174
```

input

6 1 8 7 6 3 6 5 9 6 8 8 6
output
235

Note

In the first example, you can reverse the subarray $[4, 5]$. Then $a = [2, 3, 2, 3, 1]$ and $2 \cdot 1 + 3 \cdot 3 + 2 \cdot 2 + 3 \cdot 4 + 1 \cdot 2 = 29$.

In the second example, you don't need to use the reverse operation. $13 \cdot 2 + 37 \cdot 4 = 174$.

In the third example, you can reverse the subarray $[3, 5]$. Then $a = [1, 8, 3, 6, 7, 6]$ and $1 \cdot 5 + 8 \cdot 9 + 3 \cdot 6 + 6 \cdot 8 + 7 \cdot 8 + 6 \cdot 6 = 235$.

E. Off by One

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n points on an infinite plane. The i-th point has coordinates (x_i, y_i) such that $x_i > 0$ and $y_i > 0$. The coordinates are not necessarily integer.

In one move you perform the following operations:

- choose two points a and b ($a \neq b$);
- move point a from (x_a, y_a) to either $(x_a + 1, y_a)$ or $(x_a, y_a + 1)$;
- move point b from (x_b, y_b) to either $(x_b + 1, y_b)$ or $(x_b, y_b + 1)$;
- remove points a and b.

However, the move can only be performed if there exists a line that passes through the new coordinates of a, new coordinates of b and $(0, 0)$.

Otherwise, the move can't be performed and the points stay at their original coordinates (x_a, y_a) and (x_b, y_b), respectively.

The numeration of points **does not change** after some points are removed. Once the points are removed, they can't be chosen in any later moves. Note that you have to move both points during the move, you can't leave them at their original coordinates.

What is the maximum number of moves you can perform? What are these moves?

If there are multiple answers, you can print any of them.

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of points.

The i-th of the next n lines contains four integers a_i, b_i, c_i, d_i ($1 \le a_i, b_i, c_i, d_i \le 10^9$). The coordinates of the i-th point are $x_i = \frac{a_i}{b_i}$ and $y_i = \frac{c_i}{d_i}$.

Output

In the first line print a single integer c — the maximum number of moves you can perform.

Each of the next c lines should contain a description of a move: two integers a and b ($1 \le a, b \le n$, $a \neq b$) — the points that are removed during the current move. There should be a way to move points a and b according to the statement so that there's a line that passes through the new coordinates of a, the new coordinates of b and $(0, 0)$. No removed point can be chosen in a later move.

If there are multiple answers, you can print any of them. You can print the moves and the points in the move in the arbitrary order.

Examples

input
7 4 1 5 1 1 1 1 1 3 3 3 3 1 1 4 1 6 1 1 1 5 1 4 1 6 1 1 1
output
3 1 6 2 4 5 7

input

```
4
2 1 1 1
1 1 2 1
2 1 1 2
1 2 1 2
```

output

```
1
1 2
```

input

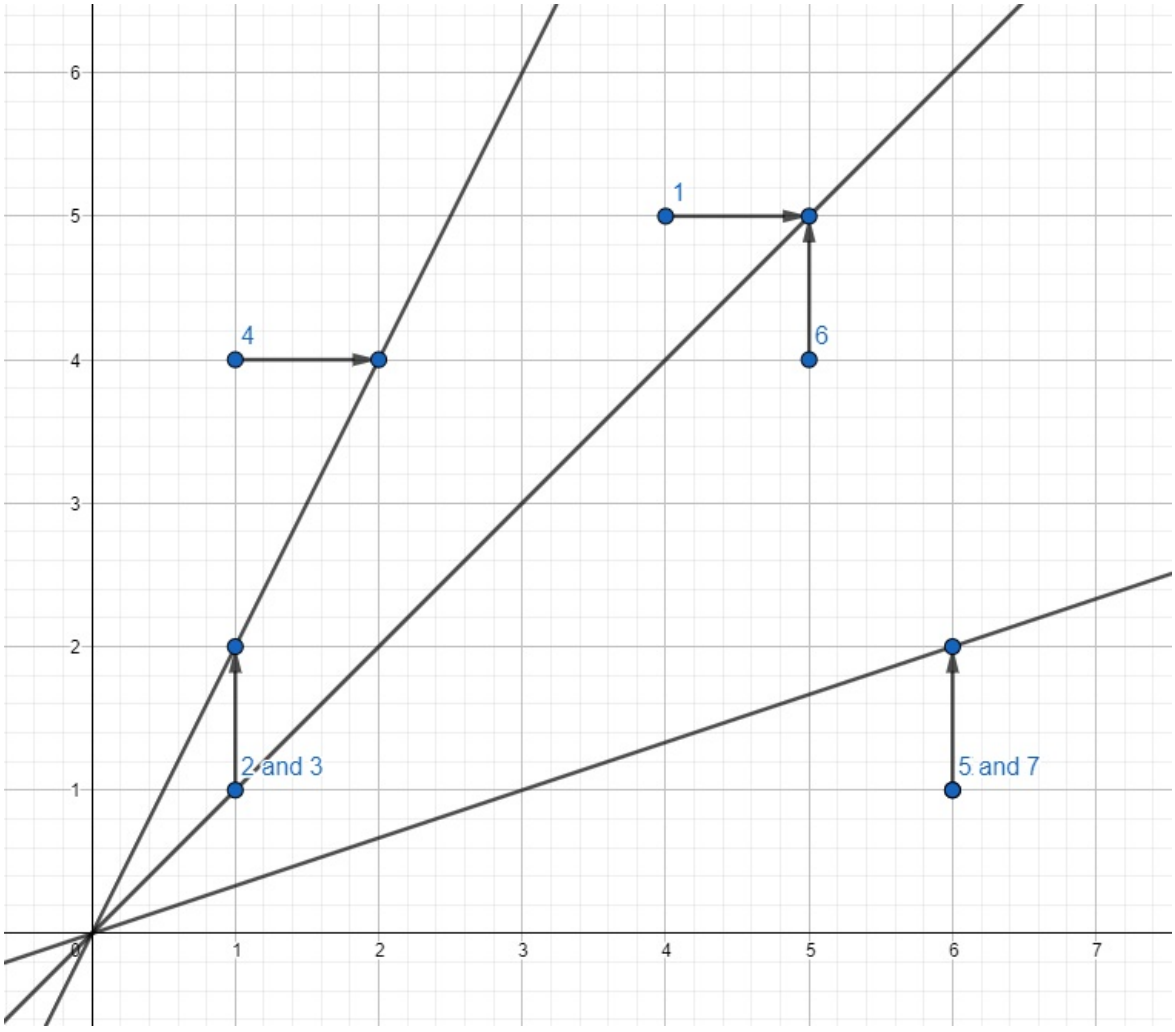
```
4
182 168 60 96
78 72 45 72
69 21 144 63
148 12 105 6
```

output

```
1
2 4
```

Note

Here are the points and the moves for the ones that get chosen for the moves from the first example:



F. Chests and Keys

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Alice and Bob play a game. Alice has got n treasure chests (the i-th of which contains a_i coins) and m keys (the j-th of which she can sell Bob for b_j coins).

Firstly, Alice puts some locks on the chests. There are m types of locks, the locks of the j-th type can only be opened with the j-th key. To put a lock of type j on the i-th chest, Alice has to pay $c_{i,j}$ dollars. Alice can put any number of different types of locks on each chest (possibly, zero).

Then, Bob buys some of the keys from Alice (possibly none, possibly all of them) and opens each chest he can (he can open a chest if he has the keys for all of the locks on this chest). Bob's profit is the difference between the total number of coins in the opened chests and the total number of coins he spends buying keys from Alice. If Bob's profit is **strictly positive** (greater than zero), he

wins the game. Otherwise, Alice wins the game.

Alice wants to put some locks on some chests so no matter which keys Bob buys, she always wins (Bob cannot get positive profit). Of course, she wants to spend the minimum possible number of dollars on buying the locks. Help her to determine whether she can win the game at all, and if she can, how many dollars she has to spend on the locks.

Input

The first line contains two integers n and m ($1 \le n, m \le 6$) — the number of chests and the number of keys, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 4$), where a_i is the number of coins in the i-th chest.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_j \le 4$), where b_j is the number of coins Bob has to spend to buy the j-th key from Alice.

Then n lines follow. The i-th of them contains m integers $c_{i,1}, c_{i,2}, \ldots, c_{i,m}$ ($1 \le c_{i,j} \le 10^7$), where $c_{i,j}$ is the number of dollars Alice has to spend to put a lock of the j-th type on the i-th chest.

Output

If Alice cannot ensure her victory (no matter which locks she puts on which chests, Bob always has a way to gain positive profit), print -1.

Otherwise, print one integer — the minimum number of dollars Alice has to spend to win the game regardless of Bob's actions.

Examples

input
2 3 3 3 1 1 4 10 20 100 20 15 80
output
205

input
2 3 3 3 2 1 4 10 20 100 20 15 80
output
110

input
2 3 3 4 1 1 4 10 20 100 20 15 80
output
-1

Note

In the first example, Alice should put locks of types 1 and 3 on the first chest, and locks of type 2 and 3 on the second chest.

In the second example, Alice should put locks of types 1 and 2 on the first chest, and a lock of type 3 on the second chest.