Codeforces Round #608 (Div. 2)

A. Suits

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

A new delivery of clothing has arrived today to the clothing store. This delivery consists of a ties, b scarves, c vests and d jackets.

The store does not sell single clothing items — instead, it sells suits of two types:

- a suit of the first type consists of one tie and one jacket;
- a suit of the second type consists of one scarf, one vest and one jacket.

Each suit of the first type costs e coins, and each suit of the second type costs f coins.

Calculate the maximum possible cost of a set of suits that can be composed from the delivered clothing items. Note that one item cannot be used in more than one suit (though some items may be left unused).

Input

The first line contains one integer a a — the number of ties.

The second line contains one integer b b — the number of scarves.

The third line contains one integer c c — the number of vests.

The fourth line contains one integer d d — the number of jackets.

The fifth line contains one integer e e — the cost of one suit of the first type.

The sixth line contains one integer f f — the cost of one suit of the second type.

Output

Print one integer — the maximum total cost of some set of suits that can be composed from the delivered items.

Examples

input
4 5 6 3 1 2
output
6

input
12 11 13 20 4 6
output
102

input
17 14 5 21 15 17
output
325

Note

It is possible to compose three suits of the second type in the first example, and their total cost will be . Since all jackets will be

used, it's impossible to add anything to this set.

The best course of action in the second example is to compose nine suits of the first type and eleven suits of the second type. The total cost is .

B. Blocks

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

There are n blocks arranged in a row and numbered from left to right, starting from one. Each block is either black or white.

You may perform the following operation zero or more times: choose two **adjacent** blocks and invert their colors (white block becomes black, and vice versa).

You want to find a sequence of operations, such that they make all the blocks having the same color. You **don't have** to minimize the number of operations, but it should not exceed n. If it is impossible to find such a sequence of operations, you need to report it.

Input

The first line contains one integer n (n) — the number of blocks.

The second line contains one string s consisting of n characters, each character is either "W" or "B". If the i-th character is "W", then the i-th block is white. If the i-th character is "B", then the i-th block is black.

Output

If it is impossible to make all the blocks having the same color, print .

Otherwise, print an integer k (k n) — the number of operations. Then print k integers p p p_k p_j n , where p_j is the position of the left block in the pair of blocks that should be affected by the j-th operation.

If there are multiple answers, print any of them.

Examples

input
8 BWWWWWWB
output
3 6 2 4

input
4 BWBB
output
-1

input
5 WWWWW
output
0

input
3 BWB
output
2 2 1

Note

In the first example, it is possible to make all blocks black in operations. Start with changing blocks and , so the sequence is "BWWWWBBB". Then change blocks and , so the sequence is "BBBWWBB". And finally, change blocks and , so all blocks are black.

It is impossible to make all colors equal in the second example.

All blocks are already white in the third example.

In the fourth example it is possible to make all blocks white in two operations: first operation is to change blocks and (so the sequence is "BBW"), and then change blocks and (so all blocks are white).

C. Shawarma Tent

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

The map of the capital of Berland can be viewed on the infinite coordinate plane. Each point with integer coordinates contains a building, and there are streets connecting every building to four neighbouring buildings. All streets are parallel to the coordinate axes.

The main school of the capital is located in s_x s_y . There are n students attending this school, the i-th of them lives in the house located in x_i y_i . It is possible that some students live in the same house, but no student lives in s_x s_y .

After classes end, each student walks from the school to his house along one of the shortest paths. So the distance the i-th student goes from the school to his house is s_x x_i s_y y_i .

The Provision Department of Berland has decided to open a shawarma tent somewhere in the capital (at some point with integer coordinates). It is considered that the i-th student will buy a shawarma if at least one of the shortest paths from the school to the i-th student's house goes through the point where the shawarma tent is located. It is forbidden to place the shawarma tent at the point where the school is located, but the coordinates of the shawarma tent may coincide with the coordinates of the house of some student (or even multiple students).

You want to find the maximum possible number of students buying shawarma and the optimal location for the tent itself.

Input

The first line contains three integers n, s_x, s_y (n , s_x s_y) — the number of students and the coordinates of the school, respectively.

Then n lines follow. The i-th of them contains two integers x_i, y_i (x_i y_i) — the location of the house where the i-th student lives. Some locations of houses may coincide, but no student lives in the same location where the school is situated.

Output

The output should consist of two lines. The first of them should contain one integer c — the maximum number of students that will buy shawarmas at the tent.

The second line should contain two integers p_x and p_y — the coordinates where the tent should be located. If there are multiple answers, print any of them. Note that each of p_x and p_y should be not less than and not greater than .

Examples

input
4 3 2 1 3 4 2 5 1 4 1
output
3 4 2

input
3 100 100 0 0 0 0 100 200
output
2 99 100

input
7 10 12 5 6 20 23 15 4 16 5 4 54 12 1 4 15
output
4 10 11

Note

In the first example, If we build the shawarma tent in , then the students living in , and will visit it.

In the second example, it is possible to build the shawarma tent in , then both students living in will visit it.

D. Portals

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

You play a strategic video game *(yeah, we ran out of good problem legends)*. In this game you control a large army, and your goal is to conquer n castles of your opponent.

Let's describe the game process in detail. Initially you control an army of k warriors. Your enemy controls n castles; to conquer the i-th castle, you need at least a_i warriors (you are so good at this game that you don't lose any warriors while taking over a castle, so your army stays the same after the fight). After you take control over a castle, you recruit new warriors into your army — formally, after you capture the i-th castle, b_i warriors join your army. Furthermore, after capturing a castle (or later) you can defend it: if you leave at least one warrior in a castle, this castle is considered *defended*. Each castle has an importance parameter c_i, and your total score is the sum of importance values over all defended castles. There are two ways to defend a castle:

- if you are currently in the castle i, you may leave one warrior to defend castle i;
- there are m one-way portals connecting the castles. Each portal is characterised by two numbers of castles u and v *(for each portal holds u v)*. A portal can be used as follows: if you are currently in the castle u, you may send one warrior to defend castle v.

Obviously, when you order your warrior to defend some castle, he leaves your army.

You capture the castles in fixed order: you have to capture the first one, then the second one, and so on. After you capture the castle i (but only before capturing castle i) you may recruit new warriors from castle i, leave a warrior to defend castle i, and use any number of portals leading from castle i to other castles having smaller numbers. As soon as you capture the next castle, these actions for castle i won't be available to you.

If, during some moment in the game, you don't have enough warriors to capture the next castle, you lose. Your goal is to maximize the sum of importance values over all defended castles (note that you may hire new warriors in the last castle, defend it and use portals leading from it even after you capture it — your score will be calculated afterwards).

Can you determine an optimal strategy of capturing and defending the castles?

Input

The first line contains three integers n, m and k (n , m $\frac{n\ n}{}$, k) — the number of castles, the number of portals and initial size of your army, respectively.

Then n lines follow. The i-th line describes the i-th castle with three integers a_i, b_i and c_i (a_i b_i c_i) — the number of warriors required to capture the i-th castle, the number of warriors available for hire in this castle and its importance value.

Then m lines follow. The i-th line describes the i-th portal with two integers u_i and v_i (v_i u_i n), meaning that the portal leads from the castle u_i to the castle v_i. There are no two same portals listed.

It is guaranteed that the size of your army won't exceed under any circumstances (i. e. k $\sum_{i}^{n} b_i$).

Output

If it's impossible to capture all the castles, print one integer .

Otherwise, print one integer equal to the maximum sum of importance values of defended castles.

Examples

input
4 3 7 7 4 17 3 0 8 11 2 0 13 3 5 3 1 2 1 4 3
output
5

input
4 3 7 7 4 17 3 0 8 11 2 0 13 3 5 3 1 2 1 4 1

output
22

input
4 3 7 7 4 17 3 0 8 11 2 0 14 3 5 3 1 2 1 4 3

output
-1

Note

The best course of action in the first example is as follows:

1. capture the first castle;
2. hire warriors from the first castle, your army has warriors now;
3. capture the second castle;
4. capture the third castle;
5. hire warriors from the third castle, your army has warriors now;
6. capture the fourth castle;
7. leave one warrior to protect the fourth castle, your army has warriors now.

This course of action (and several other ones) gives as your total score.

The best course of action in the second example is as follows:

1. capture the first castle;
2. hire warriors from the first castle, your army has warriors now;
3. capture the second castle;
4. capture the third castle;
5. hire warriors from the third castle, your army has warriors now;
6. capture the fourth castle;
7. leave one warrior to protect the fourth castle, your army has warriors now;
8. send one warrior to protect the first castle through the third portal, your army has warriors now.

This course of action (and several other ones) gives as your total score.

In the third example it's impossible to capture the last castle: you need warriors to do so, but you can accumulate no more than without capturing it.

E. Common Number

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

At first, let's define function $f\ x$ as follows:

$$f\ x \qquad \frac{x}{} \qquad x \qquad x$$

We can see that if we choose some value v and will apply function f to it, then apply f to $f\ v$, and so on, we'll eventually get . Let's write down all values we get in this process in a list and denote this list as $path\ v$. For example, $path$, $path$, $path$.

Let's write all lists $path\ x$ for every x from to n. The question is next: what is the maximum value y such that y is contained in at least k different lists $path\ x$?

Formally speaking, you need to find maximum y such that x x n y $path\ x$ k.

Input

The first line contains two integers n and k (k n).

Output

Print the only integer — the maximum value that is contained in at least k paths.

Examples

input

11 3
output
5

input
11 6
output
4

input
20 20
output
1

input
14 5
output
6

input
1000000 100
output
31248

Note

In the first example, the answer is , since occurs in $path$, $path$ and $path$.

In the second example, the answer is , since occurs in $path$, $path$, $path$, $path$, $path$ and $path$.

In the third example n k, so the answer is , since is the only number occuring in all paths for integers from to .

F. Divide The Students

time limit per test: 8 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Recently a lot of students were enrolled in Berland State University. All students were divided into groups according to their education program. Some groups turned out to be too large to attend lessons in the same auditorium, so these groups should be divided into two subgroups. Your task is to help divide the first-year students of the computer science faculty.

There are t new groups belonging to this faculty. Students have to attend classes on three different subjects — maths, programming and P. E. All classes are held in different places according to the subject — maths classes are held in auditoriums, programming classes are held in computer labs, and P. E. classes are held in gyms.

Each group should be divided into two subgroups so that there is enough space in every auditorium, lab or gym for all students of the subgroup. For the first subgroup of the i-th group, maths classes are held in an auditorium with capacity of a_i students; programming classes are held in a lab that accomodates up to b_i students; and P. E. classes are held in a gym having enough place for c_i students. Analogically, the auditorium, lab and gym for the second subgroup can accept no more than a_i , b_i and c_i students, respectively.

As usual, some students skip some classes. Each student considers some number of subjects (from to) to be useless — that means, he skips all classes on these subjects (and attends all other classes). This data is given to you as follows — the i-th group consists of:

1. d_i students which attend all classes;
2. d_i students which attend all classes, except for P. E.;
3. d_i students which attend all classes, except for programming;
4. d_i students which attend only maths classes;
5. d_i students which attend all classes, except for maths;
6. d_i students which attend only programming classes;
7. d_i students which attend only P. E.

There is one more type of students — those who don't attend any classes at all (but they, obviously, don't need any place in auditoriums, labs or gyms, so the number of those students is insignificant in this problem).

Your task is to divide each group into two subgroups so that every auditorium (or lab, or gym) assigned to each subgroup has enough place for all students from this subgroup attending the corresponding classes (if it is possible). Each student of the i-th

group should belong to exactly one subgroup of the i-th group; it is forbidden to move students between groups.

Input

The first line contains one integer t (t) — the number of groups.

Then the descriptions of groups follow. The description of the i-th group consists of three lines:

- the first line contains three integers a_i , b_i and c_i (a_i b_i c_i) — the capacity of the auditorium, lab and gym assigned to the first subgroup of the i-th group, respectively;
- the second line contains three integers a_i , b_i and c_i (a_i b_i c_i) — the capacity of the auditorium, lab and gym assigned to the second subgroup of the i-th group, respectively;
- the third line contains integers d_i , d_i , ..., d_i ($d_{i\,j}$) — the number of students belonging to each of the seven aforementioned types in the i-th group. **It is not guaranteed** that the sum of these values is positive — a group can consist entirely of students that don't attend classes at all.

It is guaranteed that the total number of students in all groups is not greater than .

Output

For each group, print the result of its division as follows:

- if it is impossible to divide the group, print one integer ;
- otherwise print seven integers f_i , f_i , ..., f_i ($f_{i\,j}$ $d_{i\,j}$) — the number of students the first, second, ..., seventh type in the first subgroup of the i-th group (all other students will be assigned to the second subgroup). If there are multiple answers, print any of them.

Example

input
3 9 4 13 1 10 3 1 2 3 4 5 6 7 9 4 13 1 10 3 2 1 3 4 5 6 7 1 2 3 4 5 6 0 0 0 0 0 0 0
output
1 1 3 4 2 0 7 -1 0 0 0 0 0 0 0