A. Valera and Plates

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

Valera is a lazy student. He has m clean bowls and k clean plates.

Valera has made an eating plan for the next n days. As Valera is lazy, he will eat exactly one dish per day. At that, in order to eat a dish, he needs exactly one *clean* plate or bowl. We know that Valera can cook only two types of dishes. He can eat dishes of the first type from bowls and dishes of the second type from either bowls or plates.

When Valera finishes eating, he leaves a dirty plate/bowl behind. His life philosophy doesn't let him eat from dirty kitchenware. So sometimes he needs to wash his plate/bowl **before eating**. Find the minimum number of times Valera will need to wash a plate/bowl, if he acts optimally.

Input

The first line of the input contains three integers n, m, k ($1 \le n, m, k \le 1000$) — the number of the planned days, the number of clean bowls and the number of clean plates.

The second line contains n integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 2$). If a_i equals one, then on day i Valera will eat a first type dish. If a_i equals two, then on day i Valera will eat a second type dish.

Output

Print a single integer — the minimum number of times Valera will need to wash a plate/bowl.

Examples

input

```
3 1 1
1 2 1
```

output

```
1
```

input

```
4 3 1
1 1 1 1
```

output

```
1
```

input

```
3 1 2
2 2 2
```

output

```
0
```

input

```
8 2 2
1 2 1 2 1 2 1 2
```

output

```
4
```

Note

In the first sample Valera will wash a bowl only on the third day, so the answer is one.

In the second sample, Valera will have the first type of the dish during all four days, and since there are only three bowls, he will wash a bowl exactly once.

In the third sample, Valera will have the second type of dish for all three days, and as they can be eaten from either a plate or a bowl, he will never need to wash a plate/bowl.

B. Valera and Contest

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

Valera loves to participate in competitions. Especially in programming contests. Today he has participated in the contest with his team, consisting of n students (including Valera). This contest was an individual competition, so each student in the team solved problems individually.

After the contest was over, Valera was interested in results. He found out that:

- each student in the team scored at least l points and at most r points;
- in total, all members of the team scored exactly s_{all} points;
- the total score of the k members of the team who scored the most points is equal to exactly s_k; more formally, if $a_1, a_2, ..., a_n$ is the sequence of points earned by the team of students in the non-increasing order $(a_1 \geq a_2 \geq ... \geq a_n)$, then $s_k = a_1 + a_2 + ... + a_k$.

However, Valera did not find out exactly how many points each of n students scored. Valera asked you to recover any distribution of scores between the students of the team, such that all the conditions above are met.

Input

The first line of the input contains exactly six integers n, k, l, r, s_{all}, s_k ($1 \leq n, k, l, r \leq 1000$; $l \leq r$; $k \leq n$; $1 \leq s_k \leq s_{all} \leq 10^6$).

It's guaranteed that the input is such that the answer exists.

Output

Print exactly n integers $a_1, a_2, ..., a_n$ — the number of points each student scored. If there are multiple solutions, you can print any of them. You can print the distribution of points in any order.

Examples

input

```
5 3 1 3 13 9
```

output

```
2 3 2 3 3
```

input

```
5 3 1 3 15 9
```

output

```
3 3 3 3 3
```

C. Valera and Elections

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

The city Valera lives in is going to hold elections to the city Parliament.

The city has n districts and $n - 1$ bidirectional roads. We know that from any district there is a path along the roads to any other district. Let's enumerate all districts in some way by integers from 1 to n, inclusive. Furthermore, for each road the residents decided if it is the problem road or not. *A problem road* is a road that needs to be repaired.

There are n candidates running the elections. Let's enumerate all candidates in some way by integers from 1 to n, inclusive. If the candidate number i will be elected in the city Parliament, he will perform exactly one promise — to repair all problem roads on the way from the i-th district to the district 1, where the city Parliament is located.

Help Valera and determine the subset of candidates such that if all candidates from the subset will be elected to the city Parliament, all problem roads in the city will be repaired. If there are several such subsets, you should choose the subset consisting of the minimum number of candidates.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of districts in the city.

Then $n - 1$ lines follow. Each line contains the description of a city road as three positive integers x_i, y_i, t_i ($1 \le x_i, y_i \le n$, $1 \le t_i \le 2$) — the districts connected by the i-th bidirectional road and the road type. If t_i equals to one, then the i-th road isn't the problem road; if t_i equals to two, then the i-th road is the problem road.

It's guaranteed that the graph structure of the city is a tree.

Output

In the first line print a single non-negative number k — the minimum size of the required subset of candidates. Then on the second line print k space-separated integers $a_1, a_2, \dots a_k$ — the numbers of the candidates that form the required subset. If there are multiple solutions, you are allowed to print any of them.

Examples

input

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

output

```
1
5
```

input

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

output

```
1
3
```

input

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

output

```
4
5 4 3 2
```

D. Valera and Fools

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

One fine morning, n fools lined up in a row. After that, they numbered each other with numbers from 1 to n, inclusive. Each fool got a unique number. The fools decided not to change their numbers before the end of the fun.

Every fool has exactly k bullets and a pistol. In addition, the fool number i has probability of p_i (in percent) that he kills the fool he shoots at.

The fools decided to have several rounds of the fun. Each round of the fun looks like this: each currently living fool shoots at another living fool with the smallest number (a fool is not stupid enough to shoot at himself). All shots of the round are perfomed at one time (simultaneously). If there is exactly one living fool, he does not shoot.

Let's define a *situation* as the set of numbers of all the living fools at the some time. We say that a situation is *possible* if for some integer number j ($0 \le j \le k$) there is a nonzero probability that after j rounds of the fun this situation will occur.

Valera knows numbers $p_1, p_2, \ldots, p_n$ and k. Help Valera determine the number of distinct *possible situations*.

Input

The first line contains two integers n, k ($1 \le n, k \le 3000$) — the initial number of fools and the number of bullets for each fool.

The second line contains n integers $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le 100$) — the given probabilities (in percent).

Output

Print a single number — the answer to the problem.

Examples

input

```
3 3
50 50 50
```

output

```
7
```

input

```
1 1
100
```

output

```
1
```

input

```
2 1
100 100
```

output

```
2
```

input

```
3 3
0 0 0
```

output

```
1
```

Note

In the first sample, any situation is possible, except for situation $\{1, 2\}$.

In the second sample there is exactly one fool, so he does not make shots.

In the third sample the possible situations are $\{1, 2\}$ (after zero rounds) and the "empty" situation $\{\}$ (after one round).

In the fourth sample, the only possible situation is $\{1, 2, 3\}$.

E. Valera and Queries

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Valera loves segments. He has recently come up with one interesting problem.

The Ox axis of coordinates has n segments, the i-th segment starts in position l_i and ends in position r_i (we will mark it as $[l_i, r_i]$). Your task is to process m queries, each consists of number cnt_i and a set of cnt_i coordinates of points located on the Ox axis. The answer to the query is the number of segments, such that each of them contains at least one point from the query. Segment $[l, r]$ *contains* point q, if $l \le q \le r$.

Valera found the solution of this problem too difficult. So he asked you to help him. Help Valera.

Input

The first line contains two integers n, m ($1 \le n, m \le 3 \cdot 10^5$) — the number of segments on the axis of coordinates and the number of queries.

Next n lines contain the descriptions of the segments. The i-th line contains two positive integers l_i, r_i ($1 \le l_i \le r_i \le 10^6$) — the borders of the i-th segment.

Next m lines contain the description of the queries, one per line. Each line starts from integer cnt_i ($1 \le cnt_i \le 3 \cdot 10^5$) — the number of points in the i-th query. Then the line contains cnt_i distinct positive integers $p_1, p_2, \ldots, p_{cnt_i}$ ($1 \le p_1 < p_2 < \ldots < p_{cnt_i} \le 10^6$) — the coordinates of points in the i-th query.

It is guaranteed that the total number of points in all queries doesn't exceed $3 \cdot 10^5$.

Output

Print m non-negative integers, where the i-th number is the response to the i-th query.

Examples

input

```
3 3
1 3
4 5
6 7
3 1 4 7
2 4 5
1 8
```

output

```
3
1
0
```