Codeforces Round #145 (Div. 1, ACM-ICPC Rules)

A. Cinema

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

Overall there are m actors in Berland. Each actor has a personal identifier — an integer from 1 to m (distinct actors have distinct identifiers). Vasya likes to watch Berland movies with Berland actors, and he has k favorite actors. He watched the movie trailers for the next month and wrote the following information for every movie: the movie title, the number of actors who starred in it, and the identifiers of these actors. Besides, he managed to copy the movie titles and how many actors starred there, but he didn't manage to write down the identifiers of some actors. Vasya looks at his records and wonders which movies may be his favourite, and which ones may not be. Once Vasya learns the exact cast of all movies, his favorite movies will be determined as follows: a movie becomes favorite movie, if no other movie from Vasya's list has more favorite actors.

Help the boy to determine the following for each movie:

- whether it surely will be his favourite movie;
- whether it surely won't be his favourite movie;
- can either be favourite or not.

Input

The first line of the input contains two integers m and k ($1 \le m \le 100$, $1 \le k \le m$) — the number of actors in Berland and the number of Vasya's favourite actors.

The second line contains k distinct integers a_i ($1 \le a_i \le m$) — the identifiers of Vasya's favourite actors.

The third line contains a single integer n ($1 \le n \le 100$) — the number of movies in Vasya's list.

Then follow n blocks of lines, each block contains a movie's description. The i-th movie's description contains three lines:

- the first line contains string s_i (s_i consists of lowercase English letters and can have the length of from 1 to 10 characters, inclusive) — the movie's title,
- the second line contains a non-negative integer d_i ($1 \le d_i \le m$) — the number of actors who starred in this movie,
- the third line has d_i integers $b_{i,j}$ ($0 \le b_{i,j} \le m$) — the identifiers of the actors who star in this movie. If $b_{i,j} = 0$, than Vasya doesn't remember the identifier of the j-th actor. It is guaranteed that the list of actors for a movie doesn't contain the same actors.

All movies have distinct names. The numbers on the lines are separated by single spaces.

Output

Print n lines in the output. In the i-th line print:

- 0, if the i-th movie will surely be the favourite;
- 1, if the i-th movie won't surely be the favourite;
- 2, if the i-th movie can either be favourite, or not favourite.

Sample test(s)

input

```
5 3
1 2 3
6
firstfilm
3
0 0 0
secondfilm
4
0 0 4 5
thirdfilm
1
2
fourthfilm
1
5
fifthfilm
1
4
sixthfilm
2
1 0
```

output

```
2
```

```
2
1
1
1
2
```

input

```
5 3
1 3 5
4
jumanji
3
0 0 0
theeagle
5
1 2 3 4 0
matrix
3
2 4 0
sourcecode
2
2 4
```

output

```
2
0
1
1
```

Note

Note to the second sample:

- Movie `jumanji` can theoretically have from 1 to 3 Vasya's favourite actors.
- Movie `theeagle` has all three favourite actors, as the actor Vasya failed to remember, can only have identifier 5.
- Movie `matrix` can have exactly one favourite actor.
- Movie `sourcecode` doesn't have any favourite actors.

Thus, movie `theeagle` will surely be favourite, movies `matrix` and `sourcecode` won't surely be favourite, and movie `jumanji` can be either favourite (if it has all three favourite actors), or not favourite.

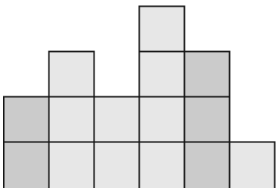
B. Fence

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

Vasya should paint a fence in front of his own cottage. The fence is a sequence of n wooden boards arranged in a single row. Each board is a 1 centimeter wide rectangle. Let's number the board fence using numbers $1, 2, ..., n$ from left to right. The height of the i-th board is h_i centimeters.

Vasya has a 1 centimeter wide brush and the paint of two colors, red and green. Of course, the amount of the paint is limited. Vasya counted the area he can paint each of the colors. It turned out that he can not paint over a square centimeters of the fence red, and he can not paint over b square centimeters green. Each board of the fence should be painted exactly one of the two colors. Perhaps Vasya won't need one of the colors.

In addition, Vasya wants his fence to look smart. To do this, he should paint the fence so as to minimize the value that Vasya called the fence *unattractiveness* value. Vasya believes that two consecutive fence boards, painted different colors, look unattractive. The *unattractiveness* value of a fence is the total length of contact between the neighboring boards of various colors. To make the fence look nice, you need to minimize the value as low as possible. Your task is to find what is the minimum unattractiveness Vasya can get, if he paints his fence completely.



The picture shows the fence, where the heights of boards (from left to right) are 2,3,2,4,3,1. The first and the fifth boards are painted red, the others are painted green. The first and the second boards have contact length 2, the fourth and fifth boards have contact length 3, the fifth and the sixth have contact length 1. Therefore, the *unattractiveness* of the given painted fence is 2+3+1=6.

Input

The first line contains a single integer n $(1 \le n \le 200)$ — the number of boards in Vasya's fence.

The second line contains two integers a and b $(0 \le a, b \le 4 \cdot 10^4)$ — the area that can be painted red and the area that can be painted green, correspondingly.

The third line contains a sequence of n integers $h_1, h_2, ..., h_n$ $(1 \le h_i \le 200)$ — the heights of the fence boards.

All numbers in the lines are separated by single spaces.

Output

Print a single number — the minimum *unattractiveness* value Vasya can get if he paints his fence completely. If it is impossible to do, print -1.

Sample test(s)

input
4 5 7 3 3 4 1
output
3

input
3 2 3 1 3 1
output
2

input
3 3 3 2 2 2
output
-1

C. Practice

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

Little time is left before Berland annual football championship. Therefore the coach of team "Losewille Rangers" decided to resume the practice, that were indefinitely interrupted for uncertain reasons. Overall there are n players in "Losewille Rangers". Each player on the team has a number — a unique integer from 1 to n. To prepare for the championship, the coach Mr. Floppe decided to spend some number of practices.

Mr. Floppe spent some long nights of his holiday planning how to conduct the practices. He came to a very complex practice system. Each practice consists of one game, all n players of the team take part in the game. The players are sorted into two teams in some way. In this case, the teams may have different numbers of players, but each team must have at least one player.

The coach wants to be sure that after the series of the practice sessions each pair of players had at least one practice, when they played in different teams. As the players' energy is limited, the coach wants to achieve the goal in the least number of practices.

Help him to schedule the practices.

Input

A single input line contains integer n ($2 \le n \le 1000$).

Output

In the first line print m — the minimum number of practices the coach will have to schedule. Then print the descriptions of the practices in m lines.

In the i-th of those lines print f_i — the number of players in the first team during the i-th practice ($1 \le f_i < n$), and f_i numbers from 1 to n — the numbers of players in the first team. The rest of the players will play in the second team during this practice. Separate numbers on a line with spaces. Print the numbers of the players in any order. If there are multiple optimal solutions, print any of them.

Sample test(s)

input
2

output
1 1 1

input
3

output
2 2 1 2 1 1

D. Merging Two Decks

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

There are two decks of cards lying on the table in front of you, some cards in these decks lay face up, some of them lay face down. You want to merge them into one deck in which each card is face down. You're going to do it in two stages.

The first stage is to merge the two decks in such a way that the relative order of the cards from the same deck doesn't change. That is, for any two different cards i and j in one deck, if card i lies above card j, then after the merge card i must also be above card j.

The second stage is performed on the deck that resulted from the first stage. At this stage, the executed operation is the turning operation. In one turn you can take a few of the top cards, turn all of them, and put them back. Thus, each of the taken cards gets turned and the order of these cards is reversed. That is, the card that was on the bottom before the turn, will be on top after it.

Your task is to make sure that all the cards are lying face down. Find such an order of merging cards in the first stage and the sequence of turning operations in the second stage, that make all the cards lie face down, and the number of turns is minimum.

Input

The first input line contains a single integer n — the number of cards in the first deck $(1 \le n \le 10^5)$.

The second input line contains n integers, separated by single spaces $a_1, a_2, ..., a_n$ $(0 \le a_i \le 1)$. Value a_i equals 0, if the i-th card is lying face down, and 1, if the card is lying face up. The cards are given in the order from the topmost one to the bottommost one.

The third input line contains integer m — the number of cards in the second deck $(1 \le m \le 10^5)$.

The fourth input line contains m integers, separated by single spaces $b_1, b_2, ..., b_m$ $(0 \le b_i \le 1)$. Value b_i equals 0, if the i-th card is lying face down, and 1, if the card is lying face up. The cards are given in the order from the topmost to the bottommost.

Output

In the first line print $n + m$ space-separated integers — the numbers of the cards in the order, in which they will lie after the first stage. List the cards from top to bottom. The cards from the first deck should match their indexes from 1 to n in the order from top to bottom. The cards from the second deck should match their indexes, increased by n, that is, numbers from $n + 1$ to $n + m$ in the order from top to bottom.

In the second line print a single integer x — the minimum number of turn operations you need to make all cards in the deck lie face down. In the third line print x integers: $c_1, c_2, ..., c_x$ $(1 \le c_i \le n + m)$, each of them represents the number of cards to take from the top of the deck to perform a turn operation. Print the operations in the order, in which they should be performed.

If there are multiple optimal solutions, print any of them. It is guaranteed that the minimum number of operations doesn't exceed $6 \cdot 10^5$.

Sample test(s)

input
3 1 0 1 4 1 1 1 1
output
1 4 5 6 7 2 3 3 5 6 7

input
5 1 1 1 1 1 5 0 1 0 1 0
output
6 1 2 3 4 5 7 8 9 10 4 1 7 8 9

E. Road Repairs

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

A country named Berland has n cities. They are numbered with integers from 1 to n. City with index 1 is the capital of the country. Some pairs of cities have monodirectional roads built between them. However, not all of them are in good condition. For each road we know whether it needs repairing or not. If a road needs repairing, then it is forbidden to use it. However, the Berland government can repair the road so that it can be used.

Right now Berland is being threatened by the war with the neighbouring state. So the capital officials decided to send a military squad to each city. The squads can move only along the existing roads, as there's no time or money to build new roads. However, some roads will probably have to be repaired in order to get to some cities.

Of course the country needs much resources to defeat the enemy, so you want to be careful with what you're going to throw the forces on. That's why the Berland government wants to repair the minimum number of roads that is enough for the military troops to get to any city from the capital, driving along good or repaired roads. Your task is to help the Berland government and to find out, which roads need to be repaired.

Input

The first line contains two space-separated integers n and m $(1 \le n, m \le 10^5)$ — the number of cities and the number of roads in Berland.

Next m lines contain three space-separated integers a_i, b_i, c_i $(1 \le a_i, b_i \le n, a_i \ne b_i, 0 \le c_i \le 1)$, describing the road from city a_i to city b_i. If c_i equals 0, than the given road is in a good condition. If c_i equals 1, then it needs to be repaired.

It is guaranteed that there is not more than one road between the cities in each direction.

Output

If even after all roads are repaired, it is still impossible to get to some city from the capital, print -1. Otherwise, on the first line print the minimum number of roads that need to be repaired, and on the second line print the numbers of these roads, separated by single spaces.

The roads are numbered starting from 1 in the order, in which they are given in the input.

If there are multiple sets, consisting of the minimum number of roads to repair to make travelling to any city from the capital possible, print any of them.

If it is possible to reach any city, driving along the roads that already are in a good condition, print 0 in the only output line.

Sample test(s)

input
```
3 2
1 3 0
3 2 1
```
output
```
1
2
```

input
```
4 4
2 3 0
3 4 0
4 1 0
4 2 1
```
output
```
-1
```

input
```
4 3
1 2 0
1 3 0
1 4 0
```
output
```
0
```

F. TorCoder

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: input.txt
output: output.txt

A boy named Leo doesn't miss a single TorCoder contest round. On the last TorCoder round number 100666 Leo stumbled over the following problem. He was given a string s, consisting of n lowercase English letters, and m queries. Each query is characterised by a pair of integers l_i, r_i $(1 \le l_i \le r_i \le n)$.

We'll consider the letters in the string numbered from 1 to n from left to right, that is, $s = s_1 s_2 \ldots s_n$.

After each query he must swap letters with indexes from l_i to r_i inclusive in string s so as to make substring (l_i, r_i) a palindrome. If there are multiple such letter permutations, you should choose the one where string (l_i, r_i) will be lexicographically minimum. If no such permutation exists, you should ignore the query (that is, not change string s).

Everybody knows that on TorCoder rounds input line and array size limits never exceed 60, so Leo solved this problem easily. Your task is to solve the problem on a little bit larger limits. Given string s and m queries, print the string that results after applying all m queries to string s.

Input

The first input line contains two integers n and m $(1 \le n, m \le 10^5)$ — the string length and the number of the queries.

The second line contains string s, consisting of n lowercase Latin letters.

Each of the next m lines contains a pair of integers l_i, r_i $(1 \le l_i \le r_i \le n)$ — a query to apply to the string.

Output

In a single line print the result of applying m queries to string s. Print the queries in the order in which they are given in the input.

Sample test(s)

input
7 2 aabcbaa 1 3 5 7
output
abacaba

input
3 2 abc 1 2 2 3
output
abc

Note

A *substring* (l_i, r_i) $1 \le l_i \le r_i \le n)$ of string $s = s_1 s_2 \ldots s_n$ of length n is a sequence of characters $s_{l_i} s_{l_i+1} \ldots s_{r_i}$.

A string is a *palindrome*, if it reads the same from left to right and from right to left.

String $x_1 x_2 \ldots x_p$ is *lexicographically smaller* than string $y_1 y_2 \ldots y_q$, if either $p < q$ and $x_1 = y_1, x_2 = y_2, \ldots, x_p = y_p$, or exists such number r $(r < p, r < q)$, that $x_1 = y_1, x_2 = y_2, \ldots, x_r = y_r$ and $x_{r+1} < y_{r+1}$.