Codeforces Round #145 (Div. 2, ACM-ICPC Rules)

A. Lefthanders and Righthanders

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

One fine October day a mathematics teacher Vasily Petrov went to a class and saw there n pupils who sat at the $\frac{n}{2}$ desks, two people at each desk. Vasily quickly realized that number n is even. Like all true mathematicians, Vasily has all students numbered from 1 to n.

But Vasily Petrov did not like the way the children were seated at the desks. According to him, the students whose numbers differ by 1, can not sit together, as they talk to each other all the time, distract others and misbehave.

On the other hand, if a righthanded student sits at the left end of the desk and a lefthanded student sits at the right end of the desk, they hit elbows all the time and distract each other. In other cases, the students who sit at the same desk, do not interfere with each other.

Vasily knows very well which students are lefthanders and which ones are righthanders, and he asks you to come up with any order that meets these two uncomplicated conditions (students do not talk to each other and do not bump their elbows). It is guaranteed that the input is such that at least one way to seat the students always exists.

Input

The first input line contains a single even integer n ($4 \le n \le 100$) — the number of students in the class. The second line contains exactly n capital English letters "L" and "R". If the i-th letter at the second line equals "L", then the student number i is a lefthander, otherwise he is a righthander.

Output

Print $\frac{n}{2}$ integer pairs, one pair per line. In the i-th line print the numbers of students that will sit at the i-th desk. The first number in the pair stands for the student who is sitting to the left, and the second number stands for the student who is sitting to the right. Separate the numbers in the pairs by spaces. If there are multiple solutions, print any of them.

Examples

input
6 LLRLLL
output
1 4 2 5 6 3

input
4 RRLL
output
3 1 4 2

B. Reading

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

Vasya is going to the Olympics in the city Ntown by train. The boy wants to read the textbook to prepare for the Olympics. He counted that he needed k hours for this. He also found that the light in the train changes every hour. The light is measured on a scale from 0 to 100, where 0 is very dark, and 100 is very light.

Vasya has a train lighting schedule for all n hours of the trip — n numbers from 0 to 100 each (the light level in the first hour, the second hour and so on). During each of those hours he will either read the whole time, or not read at all. He wants to choose k hours to read a book, not necessarily consecutive, so that the minimum level of light among the selected hours were maximum. Vasya is very excited before the upcoming contest, help him choose reading hours.

Input

The first input line contains two integers n and k ($1 \le n \le 1000, 1 \le k \le n$) — the number of hours on the train and the number of hours to read, correspondingly. The second line contains n space-separated integers a_i ($0 \le a_i \le 100$), a_i is the light level at the i-th hour.

Output

In the first output line print the minimum light level Vasya will read at. In the second line print k distinct space-separated integers $b_1, b_2, \ldots, b_k$, — the indexes of hours Vasya will read at ($1 \le b_i \le n$). The hours are indexed starting from 1. If there are multiple optimal solutions, print any of them. Print the numbers b_i in an arbitrary order.

Examples

input
5 3 20 10 30 40 10
output
20 1 3 4

input
6 5 90 20 35 40 60 100
output
35 1 3 4 5 6

Note

In the first sample Vasya should read at the first hour (light 20), third hour (light 30) and at the fourth hour (light 40). The minimum light Vasya will have to read at is 20.

C. Weather

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

Scientists say a lot about the problems of global warming and cooling of the Earth. Indeed, such natural phenomena strongly influence all life on our planet.

Our hero Vasya is quite concerned about the problems. He decided to try a little experiment and observe how outside daily temperature changes. He hung out a thermometer on the balcony every morning and recorded the temperature. He had been measuring the temperature for the last n days. Thus, he got a sequence of numbers $t_1, t_2, \ldots, t_n$, where the i-th number is the temperature on the i-th day.

Vasya analyzed the temperature statistics in other cities, and came to the conclusion that the city has no environmental problems, if first the temperature outside is negative for some non-zero number of days, and then the temperature is positive for some non-zero number of days. More formally, there must be a positive integer k ($1 \le k \le n - 1$) such that $t_1 < 0, t_2 < 0, \ldots, t_k < 0$ and $t_{k+1} > 0, t_{k+2} > 0, \ldots, t_n > 0$. In particular, the temperature should never be zero. If this condition is not met, Vasya decides that his city has environmental problems, and gets upset.

You do not want to upset Vasya. Therefore, you want to select multiple values of temperature and modify them to satisfy Vasya's condition. You need to know what the least number of temperature values needs to be changed for that.

Input

The first line contains a single integer n ($2 \le n \le 10^5$) — the number of days for which Vasya has been measuring the temperature.

The second line contains a sequence of n integers $t_1, t_2, \ldots, t_n$ ($|t_i| \le 10^9$) — the sequence of temperature values. Numbers t_i are separated by single spaces.

Output

Print a single integer — the answer to the given task.

Examples

input
4 -1 1 -2 1
output
1

input
5 0 -1 1 2 -5
output
2

Note

Note to the first sample: there are two ways to change exactly one number so that the sequence met Vasya's condition. You can either replace the first number 1 by any negative number or replace the number -2 by any positive number.

D. 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.

Examples

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.

E. Champions' League

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

In the autumn of this year, two Russian teams came into the group stage of the most prestigious football club competition in the world — the UEFA Champions League. Now, these teams have already started to play in the group stage and are fighting for advancing to the playoffs. In this problem we are interested in the draw stage, the process of sorting teams into groups.

The process of the draw goes as follows (the rules that are described in this problem, are somehow simplified compared to the real life). Suppose n teams will take part in the group stage (n is divisible by four). The teams should be divided into groups of four. Let's denote the number of groups as m ($m = \frac{n}{4}$). Each team has a rating — an integer characterizing the team's previous achievements. The teams are sorted by the rating's decreasing (no two teams have the same rating).

After that four "baskets" are formed, each of which will contain m teams: the first m teams with the highest rating go to the first basket, the following m teams go to the second one, and so on.

Then the following procedure repeats $m - 1$ times. A team is randomly taken from each basket, first from the first basket, then from the second, then from the third, and at last, from the fourth. The taken teams form another group. After that, they are **removed** from their baskets.

The four teams remaining in the baskets after $(m - 1)$ such procedures are performed, form the last group.

In the real draw the random selection of teams from the basket is performed by people — as a rule, the well-known players of the past. As we have none, we will use a random number generator, which is constructed as follows. Its parameters are four positive integers x, a, b, c. Every time there is a call to the random number generator, it produces the following actions:

- calculates $y = (x \cdot a + b) \mod c$;
- replaces parameter x by value y (assigns $x := y$);
- returns x as another random number.

Operation $\mod$ means taking the remainder after division: $10 \mod 7 = 3, 26 \mod 2 = 0$.

A random number generator will be used in the draw as follows: each time we need to randomly choose a team from the basket, it will generate a random number k. The teams that yet remain in the basket are considered numbered with consecutive integers from 0 to $s - 1$, in the order of decreasing rating, where s is the current size of the basket. Then a team number $k \mod s$ is taken from the basket.

Given a list of teams and the parameters of the random number generator, determine the result of the draw.

Input

The first input line contains integer n ($4 \le n \le 64$, n is divisible by four) — the number of teams that take part in the sorting. The second line contains four space-separated integers x, a, b, c ($1 \le x, a, b, c \le 1000$) — the parameters of the random number generator. Each of the following n lines describes one team. The description consists of the name of the team and its rating, separated by a single space. The name of a team consists of uppercase and lowercase English letters and has length from 1 to 20 characters. A team's rating is an integer from 0 to 1000. All teams' names are distinct. All team's ratings are also distinct.

Output

Print the way the teams must be sorted into groups. Print the groups in the order, in which they are formed in the sorting. Number the groups by consecutive uppercase English letters, starting from letter 'A'. Inside each group print the teams' names one per line, in the order of decreasing of the teams' rating. See samples for a better understanding of the output format.

Examples

input
8 1 3 1 7 Barcelona 158 Milan 90 Spartak 46 Anderlecht 48 Celtic 32 Benfica 87 Zenit 79 Malaga 16
output
Group A: Barcelona Benfica Spartak Celtic Group B:

```
Milan
Zenit
Anderlecht
Malaga
```

Note

In the given sample the random number generator will be executed four times:

- $(1 \cdot 3 + 1) \mod 7 = 4$,
- $(4 \cdot 3 + 1) \mod 7 = 6$,
- $(6 \cdot 3 + 1) \mod 7 = 5$,
- $(5 \cdot 3 + 1) \mod 7 = 2$.

F. 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 $1, 2, \ldots, n$ 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 $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 200$) — 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.

Examples

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

G. 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.

Examples

input
2

output
1 1 1

input
3

output
2 2 1 2 1 1

H. 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 $a_1, a_2, \ldots, a_n$ $(0 \le a_i \le 1)$. 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 $b_1, b_2, \ldots, b_m$ $(0 \le b_i \le 1)$. 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: $c_1, c_2, \ldots, c_x$ $(1 \le c_i \le n + m)$, 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 $6 \cdot 10^5$.

Examples

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