A. Elevator

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

A sky scraper with 1000 floors has been built in the city of N. It has modern superfast elevators to help to travel from one floor to another. Each elevator has two doors, the front one and the back one. If one goes in through the front door, he goes out through the back one and vice versa. The elevator has two rails numbered with numbers 1 and 2. Rail 1 is located to the left of the entrance to the front door (or correspondingly, to the right of the entrance to the back door). Rail 2 is located opposite it, to the right of the entrance to the front door and to the left of the entrance to the back door. We know that each person in the city of N holds at a rail with the strongest hand.

One day a VIP person visited the city and of course, he took a look at the skyscraper and took a ride in the elevator. We know the door through which he entered and the rail he was holding at. Now we need to determine as soon as possible whether he is left-handed or right-handed.

Input

The first line indicates the door through which the very important person entered the elevator. It contains "`front`" if the person enters the elevator through the front door and "`back`" if he entered the elevator through the back door. The second line contains integer a ($1 \le a \le 2$) which denotes the number of the rail at which the person was holding.

Output

Print character "`R`" if the VIP is right-handed or "`L`" if he is left-handed.

Examples

input

```
front
1
```

output

```
L
```

B. Quiz League

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

A team quiz game called "What? Where? When?" is very popular in Berland. The game is centered on two teams competing. They are the team of six Experts versus the team of the Audience. A person from the audience asks a question and the experts are allowed a minute on brainstorming and finding the right answer to the question. All it takes to answer a typical question is general knowledge and common logic. The question sent be the audience are in envelops lain out in a circle on a round table. Each envelop is marked by the name of the asker's town. Each question is positioned in a separate sector. In the centre of the table is a spinning arrow. Thus, the table rather resembles a roulette table with no ball but with a spinning arrow instead. The host sets off the spinning arrow to choose a question for the experts: when the arrow stops spinning, the question it is pointing at is chosen. If the arrow points at the question that has already been asked, the host chooses the next unanswered question in the clockwise direction. Your task is to determine which will be the number of the next asked question if the arrow points at sector number k.

Input

The first line contains two positive integers n and k ($1 \le n \le 1000$ and $1 \le k \le n$) — the numbers of sectors on the table and the number of the sector where the arrow is pointing. The second line contains n numbers: $a_i = 0$ if the question from sector i has already been asked and $a_i = 1$ if the question from sector i hasn't been asked yet ($1 \le i \le n$). The sectors are given in the clockwise order, the first sector follows after the n-th one.

Output

Print the single number — the number of the sector containing the question the experts will be asked. It is guaranteed that the answer exists, that is that not all the questions have already been asked.

Examples

input

```
5 5
0 1 0 1 0
```

output

```
2
```

input

```
2 1
1 1
```

output

```
1
```

C. Winnie-the-Pooh and honey

time limit per test: 1 second

memory limit per test: 256 megabytes

input: input.txt

output: output.txt

As we all know, Winnie-the-Pooh just adores honey. Ones he and the Piglet found out that the Rabbit has recently gotten hold of an impressive amount of this sweet and healthy snack. As you may guess, Winnie and the Piglet asked to come at the Rabbit's place. Thus, there are n jars of honey lined up in front of Winnie-the-Pooh, jar number i contains a_i kilos of honey. Winnie-the-Pooh eats the honey like that: each time he chooses a jar containing most honey. If the jar has less that k kilos of honey or if Winnie-the-Pooh has already eaten from it three times, he gives the jar to Piglet. Otherwise he eats exactly k kilos of honey from the jar and puts it back. Winnie does so until he gives all jars to the Piglet. Count how much honey Piglet will overall get after Winnie satisfies his hunger.

Input

The first line contains two integers n and k ($1 \le n \le 100, 1 \le k \le 100$). The second line contains n integers a_1, a_2, ..., a_n, separated by spaces ($1 \le a_i \le 100$).

Output

Print a single number — how many kilos of honey gets Piglet.

Examples

input

```
3 3
15 8 10
```

output

```
9
```

D. Three Sons

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

Three sons inherited from their father a rectangular corn fiend divided into $n \times m$ squares. For each square we know how many tons of corn grows on it. The father, an old farmer did not love all three sons equally, which is why he bequeathed to divide his field into three parts containing A, B and C tons of corn.

The field should be divided by two parallel lines. The lines should be parallel to one side of the field and to each other. The lines should go strictly between the squares of the field. Each resulting part of the field should consist of at least one square.

Your task is to find the number of ways to divide the field as is described above, that is, to mark two lines, dividing the field in three parts so that on one of the resulting parts grew A tons of corn, B on another one and C on the remaining one.

Input

The first line contains space-separated integers n and m — the sizes of the original ($1 \le n, m \le 50, max(n, m) \ge 3$). Then the field's description follows: n lines, each containing m space-separated integers c_{ij}, ($0 \le c_{ij} \le 100$) — the number of tons of corn each square contains. The last line contains space-separated integers A, B, C ($0 \le A, B, C \le 10^6$).

Output

Print the answer to the problem: the number of ways to divide the father's field so that one of the resulting parts contained A tons of corn, another one contained B tons, and the remaining one contained C tons. If no such way exists, print 0.

Examples

input

```
3 3
1 1 1
1 1 1
1 1 1
3 3 3
```

output

```
2
```

input

```
2 5
1 1 1 1 1
2 2 2 2 2
3 6 6
```

output

```
3
```

input

```
3 3
1 2 3
3 1 2
2 3 1
5 6 7
```

output

```
0
```

Note

The lines dividing the field can be horizontal or vertical, but they should be parallel to each other.

E. Put Knight!

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

Petya and Gena play a very interesting game "Put a Knight!" on a chessboard $n \times n$ in size. In this game they take turns to put chess pieces called "knights" on the board so that no two knights could threat each other. A knight located in square (r, c) can threat squares $(r - 1, c + 2)$, $(r - 1, c - 2)$, $(r + 1, c + 2)$, $(r + 1, c - 2)$, $(r - 2, c + 1)$, $(r - 2, c - 1)$, $(r + 2, c + 1)$ and $(r + 2, c - 1)$ (some of the squares may be located outside the chessboard). The player who can't put a new knight during his move loses. Determine which player wins considering that both players play optimally well and Petya starts.

Input

The first line contains integer T ($1 \le T \le 100$) — the number of boards, for which you should determine the winning player. Next T lines contain T integers n_i ($1 \le n_i \le 10000$) — the sizes of the chessboards.

Output

For each $n_i \times n_i$ board print on a single line "0" if Petya wins considering both players play optimally well. Otherwise, print "1".

Examples

input

```
2
2
1
```

output

```
1
0
```

F. Spiders

time limit per test: 1 second

memory limit per test: 256 megabytes

input: input.txt

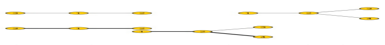
output: output.txt

One day mum asked Petya to sort his toys and get rid of some of them. Petya found a whole box of toy spiders. They were quite dear to him and the boy didn't want to throw them away. Petya conjured a cunning plan: he will glue all the spiders together and attach them to the ceiling. Besides, Petya knows that the lower the spiders will hang, the more mum is going to like it and then she won't throw his favourite toys away. Help Petya carry out the plan.

A spider consists of k beads tied together by $k - 1$ threads. Each thread connects two different beads, at that any pair of beads that make up a spider is either directly connected by a thread, or is connected via some chain of threads and beads.

Petya may glue spiders together directly gluing their beads. The length of each thread equals 1. The sizes of the beads can be neglected. That's why we can consider that gluing spiders happens by identifying some of the beads (see the picture). Besides, the construction resulting from the gluing process should also represent a spider, that is, it should have the given features.

After Petya glues all spiders together, he measures the length of the resulting toy. The distance between a pair of beads is identified as the total length of the threads that connect these two beads. The length of the resulting construction is the largest distance between all pairs of beads. Petya wants to make the spider whose length is as much as possible.



The picture two shows two spiders from the second sample. We can glue to the bead number 2 of the first spider the bead number 1 of the second spider. The threads in the spiders that form the sequence of threads of maximum lengths are highlighted on the picture.

Input

The first input file line contains one integer n ($1 \leq n \leq 100$) — the number of spiders. Next n lines contain the descriptions of each spider: integer n_i ($2 \leq n_i \leq 100$) — the number of beads, then $n_i - 1$ pairs of numbers denoting the numbers of the beads connected by threads. The beads that make up each spider are numbered from 1 to n_i.

Output

Print a single number — the length of the required construction.

Examples

input

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

output

```
2
```

input

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

output

```
4
```

input

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

output

```
7
```

G. Boom

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

Let's consider the famous game called Boom (aka Hat) with simplified rules.

There are n teams playing the game. Each team has two players. The purpose of the game is to explain the words to the teammate without using any words that contain the same root or that sound similarly.

Player j from team i $(1 \le i \le n, 1 \le j \le 2)$ is characterized by two numbers: a_{ij} and b_{ij}. The numbers correspondingly represent the skill of explaining and the skill of understanding this particular player has.

Besides, m cards are used for the game. Each card has a word written on it. The card number k $(1 \le k \le m)$ is characterized by number c_k — the complexity of the word it contains.

Before the game starts the cards are put in a deck and shuffled. Then the teams play in turns like that: the 1-st player of the 1-st team, the 1-st player of the 2-nd team, ... , the 1-st player of the n-th team, the 2-nd player of the 1-st team, ... , the 2-nd player of the n-th team, the 1-st player of the 1-st team and so on.

Each turn continues for t seconds. It goes like that: Initially the time for each turn is t. While the time left to a player is more than 0, a player takes a card from the top of the deck and starts explaining the word it has to his teammate. The time needed for the j-th player of the i-th team to explain the word from the card k to his teammate (the q-th player of the i-th team) equals $max(1, c_k - (a_{ij} + b_{iq}) - d_{ik})$ (if $j = 1,$ then $q = 2,$ else $q = 1$). The value d_{ik} is the number of seconds the i-th team has already spent explaining the word k during the previous turns. Initially, all d_{ik} equal 0. If a team manages to guess the word before the end of the turn, then the time given above is substracted from the duration of the turn, the card containing the guessed word leaves the game, the team wins one point and the game continues. If the team doesn't manage to guess the word, then the card is put at the bottom of the deck, d_{ik} increases on the amount of time of the turn, spent on explaining the word. Thus, when this team gets the very same word, they start explaining it not from the beginning, but from the point where they stopped. The game ends when words from all m cards are guessed correctly.

You are given n teams and a deck of m cards. You should determine for each team, how many points it will have by the end of the game and which words the team will have guessed.

Input

The first line contains two integers n, t $(1 \le n, t \le 100)$, which correspondingly denote the number of teams and a turn's duration.

Next n lines of the input file contain four integers each: $a_{i1}, b_{i1}, a_{i2}, b_{i2}$ $(1 \le a_{ij}, b_{ij} \le 100)$ — the skills of the first and the second player of the i-th team. The teams are given in the order in which they play.

The next line of the input file contains integer m $(1 \le m \le 100)$ the number of cards.

Next $2m$ lines contain the cards' descriptions. Two lines describe each card. The first line of the description contains the word that consists of no more than 20 characters. The words only contain small Latin letters. The second line of the description contains an integer c_k $(1 \le c_k \le 100)$ — the complexity of the word written on the k-th card. The cards are listed in the order in which the lie in the deck from top to bottom. The words on all cards are different.

Output

Print n lines. On the i-th line first print number s_i the number of points the i-th team wins. Then print s_i space-separated words — the words from the cards guessed by team i in the order in which they were guessed.

Examples

input

```
2 2
1 1 1 1
1 1 1 1
3
home
1
car
1
brother
1
```

output

```
2 home car
1 brother
```

input

```
2 4
1 2 2 1
2 3 2 2
4
armchair
3
quetzalcoatl
10
pilotage
5
defibrillator
7
```

output

```
2 armchair quetzalcoatl
2 pilotage defibrillator
```

H. Brevity is Soul of Wit

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

As we communicate, we learn much new information. However, the process of communication takes too much time. It becomes clear if we look at the words we use in our everyday speech.

We can list many simple words consisting of many letters: "information", "technologies", "university", "construction", "conservatoire", "refrigerator", "stopwatch", "windowsill", "electricity", "government" and so on. Of course, we can continue listing those words ad infinitum.

Fortunately, the solution for that problem has been found. To make our speech clear and brief, we should replace the initial words with those that resemble them but are much shorter. This idea hasn't been brought into life yet, that's why you are chosen to improve the situation.

Let's consider the following formal model of transforming words: we shall assume that one can use n words in a chat. For each words we shall introduce a notion of its shorter variant. We shall define **shorter variant** of an arbitrary word s as such word t, that meets the following conditions:

- it occurs in s as a **subsequence**,
- its length ranges from one to four characters.

In other words, the word t consists at least of one and at most of four characters that occur in the same order in the word s. Note that those characters do not necessarily follow in s immediately one after another. You are allowed not to shorten the initial word if its length does not exceed four characters.

You are given a list of n different words. Your task is to find a set of their shortened variants. The shortened variants of all words from the list should be different.

Input

The first line of the input file contains the only integer n $(1 \le n \le 200)$. Then n lines contain a set of different non-empty words that consist of lowercase Latin letters. The length of each word does not exceed 10 characters.

Output

If the solution exists, print in the output file exactly n lines, where the i-th line represents the shortened variant of the i-th word from the initial set. If there are several variants to solve the problem, print any of them. If there is no solution, print -1.

Examples

input

```
6
privet
spasibo
codeforces
java
marmelad
normalno
```

output

```
pret
sps
cdfs
java
mama
norm
```

input

```
5
aaa
aa
a
aaaa
aaaaa
```

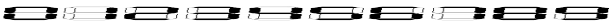
output

```
-1
```

I. Luck is in Numbers

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

Vasya has been collecting transport tickets for quite a while now. His collection contains several thousands of tram, trolleybus and bus tickets. Vasya is already fed up with the traditional definition of what a lucky ticket is. Thus, he's looking for new perspectives on that. Besides, Vasya cannot understand why all tickets are only divided into lucky and unlucky ones. He thinks that all tickets are lucky but in different degrees. Having given the matter some thought, Vasya worked out the definition of a ticket's *degree of luckiness*. Let a ticket consist of $2n$ digits. Let's regard each digit as written as is shown on the picture:



You have seen such digits on electronic clocks: seven segments are used to show digits. Each segment can either be colored or not. The colored segments form a digit. Vasya regards the digits as written in this very way and takes the right half of the ticket and puts it one the left one, so that the first digit coincides with the $n+1$-th one, the second digit coincides with the $n+2$-th one, ..., the n-th digit coincides with the $2n$-th one. For each pair of digits, put one on another, he counts the number of segments colored in both digits and summarizes the resulting numbers. The resulting value is called the *degree of luckiness* of a ticket. For example, the degree of luckiness of ticket 03 equals four and the degree of luckiness of ticket 2345 equals six.

You are given the number of a ticket containing $2n$ digits. Your task is to find among the tickets whose number exceeds the number of this ticket but also consists of $2n$ digits such ticket, whose degree of luckiness exceeds the degrees of luckiness of the given ticket. Moreover, if there are several such tickets, you should only choose the one with the smallest number.

Input

The first line contains the number of the ticket that consists of k characters ($k = 2n, 1 \le n \le 10^5$).

Output

Print the number of the sought ticket or "-1" (without the quotes) if no such ticket exists.

Examples

input

```
13
```

output

```
20
```

input

```
2345
```

output

```
2348
```

input

```
88
```

output

```
-1
```

J. Minimum Sum

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

You are given a set of n vectors on a plane. For each vector you are allowed to multiply any of its coordinates by -1. Thus, each vector $v_i = (x_i, y_i)$ can be transformed into one of the following four vectors:

- $v_i^1 = (x_i, y_i)$,
- $v_i^2 = (-x_i, y_i)$,
- $v_i^3 = (x_i, -y_i)$,
- $v_i^4 = (-x_i, -y_i)$.

You should find two vectors from the set and determine which of their coordinates should be multiplied by -1 so that the absolute value of the sum of the resulting vectors was minimally possible. More formally, you should choose two vectors v_i, v_j ($1 \le i, j \le n, i \ne j$) and two numbers k_1, k_2 ($1 \le k_1, k_2 \le 4$), so that the value of the expression $|v_i^{k_1} + v_j^{k_2}|$ were minimum.

Input

The first line contains a single integer n ($2 \le n \le 10^5$). Then n lines contain vectors as pairs of integers "x_i y_i" ($-10000 \le x_i, y_i \le 10000$), one pair per line.

Output

Print on the first line four space-separated numbers "i k_1 j k_2" — the answer to the problem. If there are several variants the absolute value of whose sums is minimum, you can print any of them.

Examples

input

```
5
-7 -3
9 0
-8 6
7 -8
4 -5
```

output

```
3 2 4 2
```

input

```
5
3 2
-4 7
-6 0
-8 4
5 1
```

output

```
3 4 5 4
```

Note

A sum of two vectors $v = (x_v, y_v)$ and $u = (x_u, y_u)$ is vector $s = v + u = (x_v + x_u, y_v + y_u)$.

An absolute value of vector $v = (x, y)$ is number $|v| = \sqrt{x^2 + y^2}$.

In the second sample there are several valid answers, such as:

(3 1 4 2), (3 1 4 4), (3 4 4 1), (3 4 4 3), (4 1 3 2), (4 1 3 4), (4 2 3 1).