Codeforces Round #568 (Div. 2)

A. Ropewalkers

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

Polycarp decided to relax on his weekend and visited to the performance of famous ropewalkers: Agafon, Boniface and Konrad.

The rope is straight and infinite in both directions. At the beginning of the performance, Agafon, Boniface and Konrad are located in positions a, b and c respectively. At the end of the performance, the distance between each pair of ropewalkers was **at least** d.

Ropewalkers can walk on the rope. In one second, only one ropewalker can change his position. Every ropewalker can change his position exactly by 1 (i. e. shift by 1 to the left or right direction on the rope). Agafon, Boniface and Konrad **can not** move at the same time (**Only one of them can move at each moment**). Ropewalkers can be at the same positions at the same time and can "walk past each other".

You should find the minimum duration (in seconds) of the performance. In other words, find the minimum number of seconds needed so that the distance between each pair of ropewalkers can be greater or equal to d.

Ropewalkers can walk to negative coordinates, due to the rope is infinite to both sides.

Input

The only line of the input contains four integers a, b, c, d ($1 \le a, b, c, d \le 10^9$). It is possible that any two (or all three) ropewalkers are in the same position at the beginning of the performance.

Output

Output one integer — the minimum duration (in seconds) of the performance.

Examples

input
```
5 2 6 3
```
output
```
2
```

input
```
3 1 5 6
```
output
```
8
```

input
```
8 3 3 2
```
output
```
2
```

input
```
2 3 10 4
```
output
```
3
```

Note

In the first example: in the first two seconds Konrad moves for 2 positions to the right (to the position 8), while Agafon and Boniface stay at their positions. Thus, the distance between Agafon and Boniface will be $|5-2|=3$, the distance between Boniface and Konrad will be $|2-8|=6$ and the distance between Agafon and Konrad will be $|5-8|=3$. Therefore, all three pairwise distances will be at least $d=3$, so the performance could be finished within 2 seconds.

B. Email from Polycarp

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

Methodius received an email from his friend Polycarp. However, Polycarp's keyboard is broken, so pressing a key on it once may cause the corresponding symbol to appear more than once (if you press a key on a regular keyboard, it prints exactly one symbol).

For example, as a result of typing the word "`hello`", the following words **could** be printed: "`hello`", "`hhhhello`", "`hheeeellllooo`", but the following **could not** be printed: "`hell`", "`helo`", "`hhllllooo`".

Note, that when you press a key, the corresponding symbol must appear (possibly, more than once). The keyboard is broken in a random manner, it means that pressing the same key you can get the different number of letters in the result.

For each word in the letter, Methodius has guessed what word Polycarp actually wanted to write, but he is not sure about it, so he asks you to help him.

You are given a list of pairs of words. For each pair, determine if the second word could be printed by typing the first one on Polycarp's keyboard.

Input

The first line of the input contains one integer n $(1 \le n \le 10^5)$ — the number of pairs to check. Further input contains n descriptions of pairs.

The first line of each description contains a single non-empty word s consisting of lowercase Latin letters. The second line of the description contains a single non-empty word t consisting of lowercase Latin letters. The lengths of both strings are not greater than 10^6.

It is guaranteed that the total length of all words s in the input is not greater than 10^6. Also, it is guaranteed that the total length of all words t in the input is not greater than 10^6.

Output

Output n lines. In the i-th line for the i-th pair of words s and t print YES if the word t could be printed by typing the word s. Otherwise, print `NO`.

Examples

input

```
4
hello
hello
hello
helloo
hello
hlllloo
hello
helo
```

output

```
YES
YES
NO
NO
```

input

```
5
aa
bb
codeforces
codeforce
polycarp
poolycarpp
aaaa
aaaab
abcdefghijklmnopqrstuvwxyz
zabcdefghijklmnopqrstuvwxyz
```

output

```
NO
NO
YES
NO
NO
```

C1. Exam in BerSU (easy version)

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

The only difference between easy and hard versions is constraints.

A session has begun at Beland State University. Many students are taking exams.

Polygraph Poligrafovich is going to examine a group of n students. Students will take the exam one-by-one in order from 1-th to n-th. Rules of the exam are following:

- The i-th student randomly chooses a ticket.
- if this ticket is too hard to the student, he doesn't answer and goes home immediately (this process is so fast that it's considered no time elapses). This student fails the exam.
- if the student finds the ticket easy, he spends exactly t_i minutes to pass the exam. After it, he immediately gets a mark and goes home.

Students take the exam in the fixed order, one-by-one, without any interruption. At any moment of time, Polygraph Poligrafovich takes the answer from one student.

The duration of the whole exam for all students is M minutes ($\max t_i \le M$), so students at the end of the list have a greater possibility to run out of time to pass the exam.

For each student i, you should count the minimum possible number of students who need to fail the exam so the i-th student has enough time to **pass** the exam.

For each student i, find the answer independently. That is, if when finding the answer for the student i_1 some student j should leave, then while finding the answer for i_2 ($i_2 > i_1$) the student j student does not have to go home.

Input

The first line of the input contains two integers n and M ($1 \le n \le 100$, $1 \le M \le 100$) — the number of students and the total duration of the exam in minutes, respectively.

The second line of the input contains n integers t_i ($1 \le t_i \le 100$) — time in minutes that i-th student spends to answer to a ticket.

It's guaranteed that all values of t_i are not greater than M.

Output

Print n numbers: the i-th number must be equal to the minimum number of students who have to leave the exam in order to i-th student has enough time to pass the exam.

Examples

input

```
7 15
1 2 3 4 5 6 7
```

output

```
0 0 0 0 0 2 3
```

input

```
5 100
80 40 40 40 60
```

output

```
0 1 1 2 3
```

Note

The explanation for the example 1.

Please note that the sum of the first five exam times does not exceed $M = 15$ (the sum is $1 + 2 + 3 + 4 + 5 = 15$). Thus, the first five students can pass the exam even if all the students before them also pass the exam. In other words, the first five numbers in the answer are 0.

In order for the 6-th student to pass the exam, it is necessary that at least 2 students must fail it before (for example, the 3-rd and 4-th, then the 6-th will finish its exam in $1 + 2 + 5 + 6 = 14$ minutes, which does not exceed M).

In order for the 7-th student to pass the exam, it is necessary that at least 3 students must fail it before (for example, the 2-nd, 5-th and 6-th, then the 7-th will finish its exam in $1 + 3 + 4 + 7 = 15$ minutes, which does not exceed M).

C2. Exam in BerSU (hard version)

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

The only difference between easy and hard versions is constraints.

If you write a solution in Python, then prefer to send it in PyPy to speed up execution time.

A session has begun at Beland State University. Many students are taking exams.

Polygraph Poligrafovich is going to examine a group of n students. Students will take the exam one-by-one in order from 1-th to n-th. Rules of the exam are following:

- The i-th student randomly chooses a ticket.
- if this ticket is too hard to the student, he doesn't answer and goes home immediately (this process is so fast that it's considered no time elapses). This student fails the exam.

- if the student finds the ticket easy, he spends exactly t_i minutes to pass the exam. After it, he immediately gets a mark and goes home.

Students take the exam in the fixed order, one-by-one, without any interruption. At any moment of time, Polygraph Poligrafovich takes the answer from one student.

The duration of the whole exam for all students is M minutes ($\max t_i \le M$), so students at the end of the list have a greater possibility to run out of time to pass the exam.

For each student i, you should count the minimum possible number of students who need to fail the exam so the i-th student has enough time to **pass** the exam.

For each student i, find the answer independently. That is, if when finding the answer for the student i_1 some student j should leave, then while finding the answer for i_2 ($i_2 > i_1$) the student j student does not have to go home.

Input

The first line of the input contains two integers n and M ($1 \le n \le 2 \cdot 10^5$, $1 \le M \le 2 \cdot 10^7$) — the number of students and the total duration of the exam in minutes, respectively.

The second line of the input contains n integers t_i ($1 \le t_i \le 100$) — time in minutes that i-th student spends to answer to a ticket.

It's guaranteed that all values of t_i are not greater than M.

Output

Print n numbers: the i-th number must be equal to the minimum number of students who have to leave the exam in order to i-th student has enough time to pass the exam.

Examples

input

```
7 15
1 2 3 4 5 6 7
```

output

```
0 0 0 0 0 2 3
```

input

```
5 100
80 40 40 40 60
```

output

```
0 1 1 2 3
```

Note

The explanation for the example 1.

Please note that the sum of the first five exam times does not exceed $M = 15$ (the sum is $1 + 2 + 3 + 4 + 5 = 15$). Thus, the first five students can pass the exam even if all the students before them also pass the exam. In other words, the first five numbers in the answer are 0.

In order for the 6-th student to pass the exam, it is necessary that at least 2 students must fail it before (for example, the 3-rd and 4-th, then the 6-th will finish its exam in $1 + 2 + 5 + 6 = 14$ minutes, which does not exceed M).

In order for the 7-th student to pass the exam, it is necessary that at least 3 students must fail it before (for example, the 2-nd, 5-th and 6-th, then the 7-th will finish its exam in $1 + 3 + 4 + 7 = 15$ minutes, which does not exceed M).

D. Extra Element

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

A sequence $a_1, a_2, \dots, a_k$ is called an arithmetic progression if for each i from 1 to k elements satisfy the condition $a_i = a_1 + c \cdot (i - 1)$ for some fixed c.

For example, these five sequences are arithmetic progressions: $[5, 7, 9, 11]$, $[101]$, $[101, 100, 99]$, $[13, 97]$ and $[5, 5, 5, 5, 5]$. And these four sequences aren't arithmetic progressions: $[3, 1, 2]$, $[1, 2, 4, 8]$, $[1, -1, 1, -1]$ and $[1, 2, 3, 3, 3]$.

You are given a sequence of integers $b_1, b_2, \dots, b_n$. Find any index j ($1 \le j \le n$), such that if you delete b_j from the sequence, you can reorder the remaining $n - 1$ elements, so that you will get an arithmetic progression. If there is no such index, output the number -1.

Input

The first line of the input contains one integer n ($2 \le n \le 2 \cdot 10^5$) — length of the sequence b. The second line contains n integers $b_1, b_2, \dots, b_n$ ($-10^9 \le b_i \le 10^9$) — elements of the sequence b.

Output

Print such index j ($1 \le j \le n$), so that if you delete the j-th element from the sequence, you can reorder the remaining elements, so that you will get an arithmetic progression. If there are multiple solutions, you are allowed to print any of them. If there is no such index, print -1.

Examples

input

```
5
2 6 8 7 4
```

output

```
4
```

input

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

output

```
1
```

input

```
4
1 2 4 8
```

output

```
-1
```

Note

Note to the first example. If you delete the 4-th element, you can get the arithmetic progression $[2, 4, 6, 8]$.

Note to the second example. The original sequence is already arithmetic progression, so you can delete 1-st or last element and you will get an arithmetical progression again.

E. Polycarp and Snakes

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

After a hard-working week Polycarp prefers to have fun. Polycarp's favorite entertainment is drawing snakes. He takes a rectangular checkered sheet of paper of size $n \times m$ (where n is the number of rows, m is the number of columns) and starts to draw snakes in cells.

Polycarp draws snakes with lowercase Latin letters. He always draws the first snake with the symbol 'a', the second snake with the symbol 'b', the third snake with the symbol 'c' and so on. All snakes have their own unique symbol. There are only 26 letters in the Latin alphabet, Polycarp is very tired and he doesn't want to invent new symbols, so the total number of drawn snakes doesn't exceed 26.

Since by the end of the week Polycarp is very tired, he draws snakes as straight lines without bends. So each snake is positioned either vertically or horizontally. Width of any snake equals 1, i.e. each snake has size either $1 \times l$ or $l \times 1$, where l is snake's length. Note that snakes can't bend.

When Polycarp draws a new snake, he can use already occupied cells for drawing the snake. In this situation, he draws the snake "over the top" and overwrites the previous value in the cell.

Recently when Polycarp was at work he found a checkered sheet of paper with Latin letters. He wants to know if it is possible to get this sheet of paper from an empty sheet by drawing some snakes according to the rules described above. If it is possible, he is interested in a way to draw snakes.

Input

The first line of the input contains one integer t ($1 \le t \le 10^5$) — the number of test cases to solve. Then t test cases follow.

The first line of the test case description contains two integers n, m ($1 \le n, m \le 2000$) — length and width of the checkered sheet of paper respectively.

Next n lines of test case description contain m symbols, which are responsible for the content of the corresponding cell on the sheet. It can be either lowercase Latin letter or symbol dot ('.'), which stands for an empty cell.

It is guaranteed that the total area of all sheets in one test doesn't exceed $4 \cdot 10^6$.

Output

Print the answer for each test case in the input.

In the first line of the output for a test case print YES if it is possible to draw snakes, so that you can get a sheet of paper from the input. If it is impossible, print NO.

If the answer to this question is positive, then print the way to draw snakes in the following format. In the next line print one integer

k ($0 \le k \le 26$) — number of snakes. Then print k lines, in each line print four integers $r_{1,i}$, $c_{1,i}$, $r_{2,i}$ and $c_{2,i}$ — coordinates of extreme cells for the i-th snake ($1 \le r_{1,i}, r_{2,i} \le n$, $1 \le c_{1,i}, c_{2,i} \le m$). Snakes should be printed in order of their drawing. If there are multiple solutions, you are allowed to print any of them.

Note that Polycarp starts drawing of snakes with an empty sheet of paper.

Examples

input

```
1
5 6
...a..
..bbb.
...a..
.cccc.
...a..
```

output

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

input

```
3
3 3
...
...
...
4 4
..c.
adda
bbcb
....
3 5
..b..
aaaaa
..b..
```

output

```
YES
0
YES
4
2 1 2 4
3 1 3 4
1 3 3 3
2 2 2 3
NO
```

input

```
2
3 3
...
.a.
...
2 2
bb
cc
```

output

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

F. Two Pizzas

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A company of n friends wants to order exactly two pizzas. It is known that in total there are 9 pizza ingredients in nature, which are denoted by integers from 1 to 9.

Each of the n friends has one or more favorite ingredients: the i-th of friends has the number of favorite ingredients equal to f_i ($1 \le f_i \le 9$) and your favorite ingredients form the sequence $b_{i1}, b_{i2}, \dots, b_{if_i}$ ($1 \le b_{it} \le 9$).

The website of CodePizza restaurant has exactly m ($m \ge 2$) pizzas. Each pizza is characterized by a set of r_j ingredients $a_{j1}, a_{j2}, \dots, a_{jr_j}$ ($1 \le r_j \le 9$, $1 \le a_{jt} \le 9$) , which are included in it, and its price is c_j.

Help your friends choose exactly two pizzas in such a way as to please the maximum number of people in the company. It is known that a person is pleased with the choice if **each** of his/her favorite ingredients is in at least one ordered pizza. If there are several ways to choose two pizzas so as to please the maximum number of friends, then choose the one that minimizes the total price of two pizzas.

Input

The first line of the input contains two integers n and m ($1 \le n \le 10^5, 2 \le m \le 10^5$) — the number of friends in the company and the number of pizzas, respectively.

Next, the n lines contain descriptions of favorite ingredients of the friends: the i-th of them contains the number of favorite ingredients f_i ($1 \le f_i \le 9$) and a sequence of distinct integers $b_{i1}, b_{i2}, \dots, b_{if_i}$ ($1 \le b_{it} \le 9$).

Next, the m lines contain pizza descriptions: the j-th of them contains the integer price of the pizza c_j ($1 \le c_j \le 10^9$), the number of ingredients r_j ($1 \le r_j \le 9$) and the ingredients themselves as a sequence of distinct integers $a_{j1}, a_{j2}, \dots, a_{jr_j}$ ($1 \le a_{jt} \le 9$).

Output

Output two integers j_1 and j_2 ($1 \le j_1, j_2 \le m$, $j_1 \ne j_2$) denoting the indices of two pizzas in the required set. If there are several solutions, output any of them. Pizza indices can be printed in any order.

Examples

input

```
3 4
2 6 7
4 2 3 9 5
3 2 3 9
100 1 7
400 3 3 2 5
100 2 9 2
500 3 2 9 5
```

output

```
2 3
```

input

```
4 3
1 1
1 2
1 3
1 4
10 4 1 2 3 4
20 4 1 2 3 4
30 4 1 2 3 4
```

output

```
1 2
```

input

```
1 5
9 9 8 7 6 5 4 3 2 1
3 4 1 2 3 4
1 4 5 6 7 8
4 4 1 3 5 7
1 4 2 4 6 8
5 4 1 9 2 8
```

output

```
2 4
```

G1. Playlist for Polycarp (easy version)

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between easy and hard versions is constraints.

Polycarp loves to listen to music, so he never leaves the player, even on the way home from the university. Polycarp overcomes the distance from the university to the house in exactly T minutes.

In the player, Polycarp stores n songs, each of which is characterized by two parameters: t_i and g_i, where t_i is the length of the song in minutes ($1 \le t_i \le 15$), g_i is its genre ($1 \le g_i \le 3$).

Polycarp wants to create such a playlist so that he can listen to music all the time on the way from the university to his home, and at the time of his arrival home, the playlist is over. Polycarp never interrupts songs and always listens to them from beginning to end. Thus, if he started listening to the i-th song, he would spend exactly t_i minutes on its listening. Polycarp also does not like when two songs of the same genre play in a row (i.e. successively/adjacently) or when the songs in his playlist are repeated.

Help Polycarpus count the number of different sequences of songs (their order matters), the total duration is exactly T, such that there are no two consecutive songs of the same genre in them and all the songs in the playlist are different.

Input

The first line of the input contains two integers n and T ($1 \le n \le 15, 1 \le T \le 225$) — the number of songs in the player and the required total duration, respectively.

Next, the n lines contain descriptions of songs: the i-th line contains two integers t_i and g_i ($1 \le t_i \le 15, 1 \le g_i \le 3$) — the duration of the i-th song and its genre, respectively.

Output

Output one integer — the number of different sequences of songs, the total length of exactly T, such that there are no two consecutive songs of the same genre in them and all the songs in the playlist are different. Since the answer may be huge, output it modulo $10^9 + 7$ (that is, the remainder when dividing the quantity by $10^9 + 7$).

Examples

input

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

output

```
6
```

input

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

output

```
2
```

input

```
4 10
5 3
2 1
3 2
5 1
```

output

```
10
```

Note

In the first example, Polycarp can make any of the 6 possible playlist by rearranging the available songs: $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$ and $[3, 2, 1]$ (indices of the songs are given).

In the second example, the first and second songs cannot go in succession (since they have the same genre). Thus, Polycarp can create a playlist in one of 2 possible ways: $[1, 3, 2]$ and $[2, 3, 1]$ (indices of the songs are given).

In the third example, Polycarp can make the following playlists: $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$, $[3, 2, 1]$, $[1, 4]$, $[4, 1]$, $[2, 3, 4]$ and $[4, 3, 2]$ (indices of the songs are given).

G2. Playlist for Polycarp (hard version)

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between easy and hard versions is constraints.

Polycarp loves to listen to music, so he never leaves the player, even on the way home from the university. Polycarp overcomes the distance from the university to the house in exactly T minutes.

In the player, Polycarp stores n songs, each of which is characterized by two parameters: t_i and g_i, where t_i is the length of the song in minutes ($1 \le t_i \le 50$), g_i is its genre ($1 \le g_i \le 3$).

Polycarp wants to create such a playlist so that he can listen to music all the time on the way from the university to his home, and at the time of his arrival home, the playlist is over. Polycarp never interrupts songs and always listens to them from beginning to end. Thus, if he started listening to the i-th song, he would spend exactly t_i minutes on its listening. Polycarp also does not like when two songs of the same genre play in a row (i.e. successively/adjacently) or when the songs in his playlist are repeated.

Help Polycarpus count the number of different sequences of songs (their order matters), the total duration is exactly T, such that there are no two consecutive songs of the same genre in them and all the songs in the playlist are different.

Input

The first line of the input contains two integers n and T ($1 \le n \le 50, 1 \le T \le 2500$) — the number of songs in the player and the required total duration, respectively.

Next, the n lines contain descriptions of songs: the i-th line contains two integers t_i and g_i ($1 \le t_i \le 50, 1 \le g_i \le 3$) — the duration of the i-th song and its genre, respectively.

Output

Output one integer — the number of different sequences of songs, the total length of exactly T, such that there are no two consecutive songs of the same genre in them and all the songs in the playlist are different. Since the answer may be huge, output it modulo $10^9 + 7$ (that is, the remainder when dividing the quantity by $10^9 + 7$).

Examples

input

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

output

```
6
```

input

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

output

```
2
```

input

```
4 10
5 3
2 1
3 2
5 1
```

output

```
10
```

Note

In the first example, Polycarp can make any of the 6 possible playlist by rearranging the available songs: $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$ and $[3, 2, 1]$ (indices of the songs are given).

In the second example, the first and second songs cannot go in succession (since they have the same genre). Thus, Polycarp can create a playlist in one of 2 possible ways: $[1, 3, 2]$ and $[2, 3, 1]$ (indices of the songs are given).

In the third example, Polycarp can make the following playlists: $[1, 2, 3]$, $[1, 3, 2]$, $[2, 1, 3]$, $[2, 3, 1]$, $[3, 1, 2]$, $[3, 2, 1]$, $[1, 4]$, $[4, 1]$, $[2, 3, 4]$ and $[4, 3, 2]$ (indices of the songs are given).