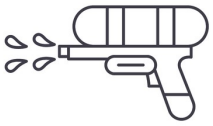
Croatian Open Competition in Informatics

Round 1, October 16th 2021

Tasks

Task	Time limit	Memory limit	Points
Ljeto	1 second	512 MiB	50
Kamenčići	1 second	512 MiB	70
Logičari	1 second	512 MiB	110
Set	1 second	512 MiB	110
Volontiranje	1 second	512 MiB	110
Total			450

Task Ljeto



Bruno and his friends are playing with water guns. They are passionate gamers, which is why this is not a regular water gun game, but is in fact very similar to video-games. They even hired a moderator for the game.

At the beginning of the game, the players are divided into two teams: pineapples and blueberries. During the game, the moderator keeps track of every time some player sprays some other player, writing down the time when it happened. Just like in video-games, the players acquire points. When a player from some team sprays someone from the opposing team, their team receives 100 points. However, if within 10 seconds the same player again sprays someone from the opposing team, this is counted as a double-spray and their team receives an additional 50 points. A player can achieve multiple double-sprays in a row, each worth an additional 50 points for their team.

Input

The first line contains the integer n ($1 \leq n \leq 100$), the number of sprays that happened during the game.

Each of the following n lines contains three integers t_i, a_i, b_i ($0 \leq t_i \leq 1000$, $1 \leq a_i, b_i \leq 8$) which denote that player a_i sprayed player b_i at time t_i (in seconds).

Labels of the players from team pineapple are positive integers from 1 to 4, while the labels of the players from team blueberry are the positive integers from 5 to 8. The players a_i and b_i are guranteed to be from different teams.

The numbers t_i are distinct and are ordered increasingly.

Output

The first and only line should contain two numbers: the total score of team pineapple and the total score of team blueberry.

Scoring

Subtask	Points	Constraints
1	10	$1 \leq n \leq 3$
2	15	No double-sprays occcur.
3	25	No additional constrains.

Examples

input

```
3
10 1 6
20 1 7
21 8 1
```

output

```
250 100
```

input

```
3
10 2 5
15 2 6
25 2 5
```

output

```
400 0
```

input

```
2
10 5 2
11 6 3
```

output

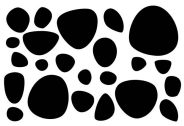
```
0 200
```

Clarification of the first example:
At seconds 10 and 20, player 1 sprayed players 6 and 7 from the other team. For each spraying the

pineapples received 100 points. As the sprays happened within 10 seconds, the team receieved an additional 50 poitns ($250 = 2 \cdot 100 + 50$). Team blueberries sprayed only one player from the opposing team, which is why they received only 100 points.

Clarification of the second example:
Player 2 achieved two double-sprays in a row, which is why team pineapple got a total of $3 \cdot 100 + 2 \cdot 50 = 400$ points.

Task Kamenčići



This summer, Antun and Branka stumbled upon a very interesting beach, which was completely covered with plastic 'pebbles' brought by the sea from the containers that fell from the cargo ships. They decided to take back with them n of these pebbles, some red and some blue. Now that autumn has arrived, they are playing with the pebbles and reminiscing about the warm summer days.

Their game proceeds as follows: in the beginning, they place the n pebbles in a row. Then, Antun and Branka make moves in turn, each time removing one of the pebbles from one of the ends of the row, until someone obtains k red pebbles, losing the game. Antun moves first and is wondering whether he could win regardless of the moves Branka makes. Please help him and write a program which will answer his question.

Input

The first line contains two integers, n and k ($1 \leq k < n \leq 350$).

The second line contains a sequence of n characters `C` or `P`, where `C` denotes a red pebble, and `P` denotes a blue pebble. The character `C` appears at least $2k - 1$ times.

Output

If Antun can win regardless of Branka's moves, you should print `DA`, otherwise print `NE`.

Scoring

Subtask	Points	Constraints
1	10	$1 \leq n \leq 20$
2	20	$1 \leq n \leq 50$
3	40	$1 \leq n \leq 350$

Examples

input	**input**	**input**
`4 1` `CCCP`	`8 2` `PCPPCCCC`	`9 1` `PPCPPCPPC`
output	**output**	**output**
`DA`	`DA`	`NE`

Clarification of the second example:
Antun can take a blue pebble from the left (`CPPCCCC`). Then, Branka has to take a red pebble.

If she takes a pebble from the left (`PPCCCC`), Antun will take the first, and Branka the second blue pebble on the left, after which only red pebbles remain and Branka will lose.

If she takes a pebble from the right (`CPPCCC`), Antun can take another pebble from the right and then Branka will again have to take another red pebble and lose.

Task Logičari

A group of perfect logicians has again received a request to be the main protagonists of a new logic puzzle. They must now agree upon which n of them should participate.

This time the logic puzzle takes place on an undirected graph with n nodes, and logically, n edges. Each edge connects two different nodes and between any two nodes there is at most one edge. Additionally, the graph is connected, which means that it is possible to go from any node to any other node via a sequence of edges. For each node there will be one logician located on that node and each logician is able to see precisely those logicians whose nodes are connected by an edge with their own node.

They already suspected that the catch might be related to their eye color, so they decided to arrange themselves so that each logician sees **exactly one** other person with blue eyes. As it ususally happens to be, none of the logicians can see their own eye color, which means that even logicians with blue eyes should see exactly one other person with blue eyes.

What is the least number of blue-eyed logicians needed to make the required arrangement?

Input

The first line contains the integer n - the number of nodes in the graph, and also the number of logicians.

The following n lines contain pairs of integers representing the edges of the graph. Each edge connects two different nodes and no edge is repeated twice in the input.

Output

If the required arrangement does not exist, in the first and only line output `-1`.

Otherwise, in the first and only line output the required least number of blue-eyed logicians.

Scoring

In every subtask, it holds that $3 \leq n \leq 100\ 000$.

Subtask	Points	Constraints
1	10	Each logician sees exactly two other logicians.
2	10	$3 \leq n \leq 20$
3	40	$3 \leq n \leq 1000$
4	50	$3 \leq n \leq 100\ 000$

Examples

input

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

output

```
2
```

input

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

output

```
-1
```

input

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

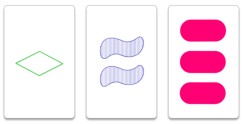
output

```
4
```

Clarification of the first example:
Blue-eyed logicians could for example be those on nodes 1 and 2.

Clarification of the second example:
If only one of the logicians has blue eyes, then he surely can't see anyone else with blue eyes. If there are two or more of them with blue eyes, then surely someone will see more than one person with blue eyes.

Task Set



In the popular card game *SET*, the player's goal is to identify a certain triplet of cards with some special properties, called a *set*. Each card shows some figures, which differ in number, shape, transparency and color.

Marin and Josip have recently bought a deck of these cards and now they can't stop playing. They've become so skilled at noticing *sets* that it soon became boring that the cards are determined by only four properties. Thus, they have decided to have fun with a generalized version of the game.

At their disposal is a deck of n **different** cards. Each card is represented by a sequence of k characters, each being one of `1`, `2` or `3`. The order of the cards in the deck does not matter.

An unordered triplet of cards is called a *set* if for each of the k positions, the three characters corresponding to the three cards are either the same or pairwise different. For example, three cards represented by `1123`, `1322` and `1221` make a *set* because all of the characters in the first and third positions are the same (`1` and `2` respectively), and the characters in the second and fourth positions are different (`1`, `2` and `3` in some order).

While looking at these n cards on the table, they started to wonder: how many unordered triplets of these n cards make a set. Write a program which will answer their question.

Input

The first line contains the integers n and k - the number of cards in the deck and the number of properties of a single card, respectively.

Each of the following n lines contains a sequence of k characters representing a card. Each character is one of `1`, `2` or `3`. Different lines contain different sequences of characters.

Output

In the only line, print the number of unordered triplets which form a *set*.

Scoring

In every subtask, it holds that $1 \leq k \leq 12$ i $1 \leq n \leq 3^k$.

Subtask	Points	Constraints
1	10	$1 \leq k \leq 5$
2	30	$1 \leq k \leq 7$
3	70	$1 \leq k \leq 12$

Examples

input

```
3 4
1123
1322
1221
```

output

```
1
```

input

```
2 2
11
22
```

output

```
0
```

input

```
5 3
111
222
333
123
132
```

output

```
2
```

Clarification of the third example:
The two *sets* are 111, 222, 333 and 111, 123 i 132.

Task Volontiranje

It is not so well known that in his free time, Mr. Malnar cotributes to the community by volunteering. That's right! He is quite active at the volunteering center for informatics problems.

Recently, he got a call from the center and, as it turns out, there is a shortage of increasing sequences. Mr. Malnar, who is always willing to help, responded readily. Namely, he keeps an array of n integers for this very purpose. All of the integers from 1 to n appear exactly once in his array.

Mr. Malnar will select a few increasing subsequences from his array, which he will donate to the center, and he will keep the rest of the numbers for some other time. It must be noted that each number from the array can be used for at most one subsequence, that is, the selected subsequences have distinct indices.

A subsequence is defined as any sequence obtained from the array by deleting some (maybe none) of the elements, while preserving the order of the remaining elements. A subsequence is said to be increasing if every number in the subsequence (except for the first one) is larger than the previous one.

Because Mr. Malnar is very generous, he wants all of the sequences he donates to be longest increasing subsequences. In other words, if l is the length of the longest increasing subsequence of the staring array, Mr. Malnar will select a couple of disjoint increasing subsequences, each of length l.

Mr. Malnar wants to donate as many subsequences as possible. For a given array of Mr. Malnar, print the maximum number of subsequences which comprise his selection and provide an example of such a selection.

Input

The first line contains the integer n - the size of the array.

The second line contains n numbers p_i ($1 \leq p_i \leq n$) which represent the elements of the array. Each positive integer j between 1 and n appears exactly once in the array.

Output

In the first line, print the largest possible number of subsequences in Mr. Malnar's selection, as well as the length of the longest increasing subsequence.

In each of the following lines print one of the subsequences from such a selection. A subsequence is defined by a list of position, i.e. **indices** of the array, which should be **sorted in increasing order**.

The printed subsequences should be **increasing**, **disjoint** and have the **same length** - the length of the longest increasing subsequence.

The relative order of the subsequences does not matter. Also, if there is more than one solution, you can print any.

Scoring

In every subtask, it holds that $1 \leq n \leq 1\ 000\ 000$.

Subtask	Points	Constraints
1	10	$1 \leq n \leq 15$
2	40	$1 \leq n \leq 1000$
4	60	$1 \leq n \leq 1\ 000\ 000$

Examples

input

```
3
1 2 3
```

output

```
1 3
1 2 3
```

input

```
4
4 3 2 1
```

output

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

input

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

output

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

Clarification of the third example:
The length of the longest increasing subsequence is 3. The subsequence determined by indices 1, 3, and 5 (or, values 2, 6 and 7) is increasing. The subsequence determined by indices 2, 6 and 7 (or, values 1, 3 and 4) is also incrasing. The two subsequences have no elements in common and they also have maximum lenght, which is why this is a valid selection. It is not possible to choose more than two of such subsequences.