Task	BARD	TETRIS	FIREFLY	CIRCLE
Input	standard input (keyboard)			
Output	standard output (screen)			
Time limit (per test)	1 second			
Memory limit (per test)	32 MB (heap) + 8 MB (stack)			
Total points	**50**	**70**	**80**	**100**
	300			

BARD

Every evening villagers in a small village gather around a big fire and sing songs.

A prominent member of the community is the bard. Every evening, if the bard is present, he sings a brand **new song** that no villager has heard before, and **no other song** is sung that night. In the event that the bard is not present, other villagers sing without him and exchange **all songs that they know.**

Given the list of villagers present for E consecutive evenings, output all villagers that know **all** songs sung during that period.

Input

The first line of input contains an integer N, 1 ≤ N ≤ 100, the number of villagers. The villagers are numbered 1 to N. Villager number 1 is the bard.

The second line contains an integer E, 1 ≤ E ≤ 50, the number of evenings.

The next E lines contain the list of villagers present on each of the E evenings. Each line begins with a positive integer K, 2 ≤ K ≤ N, the number of villagers present that evening, followed by K positive integers separated by spaces representing the villagers.

No villager will appear twice in one night and the bard will appear at least once across all nights.

Output

Output all villagers that know all songs, including the bard, one integer per line in ascending order.

Sample test data

input

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

output

```
1
2
4
```

input

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

output

```
1
2
6
8
```

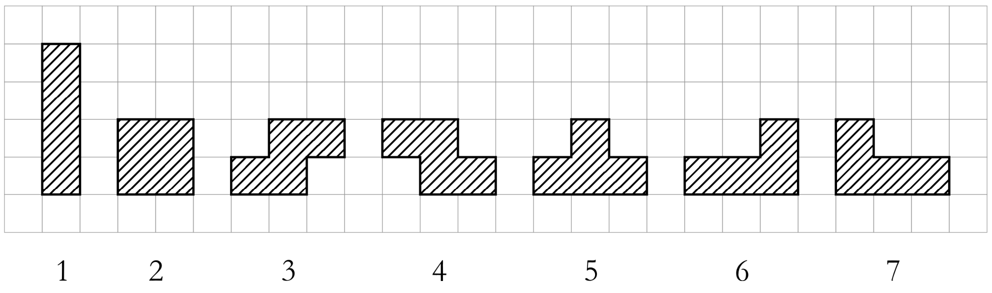
input

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

output

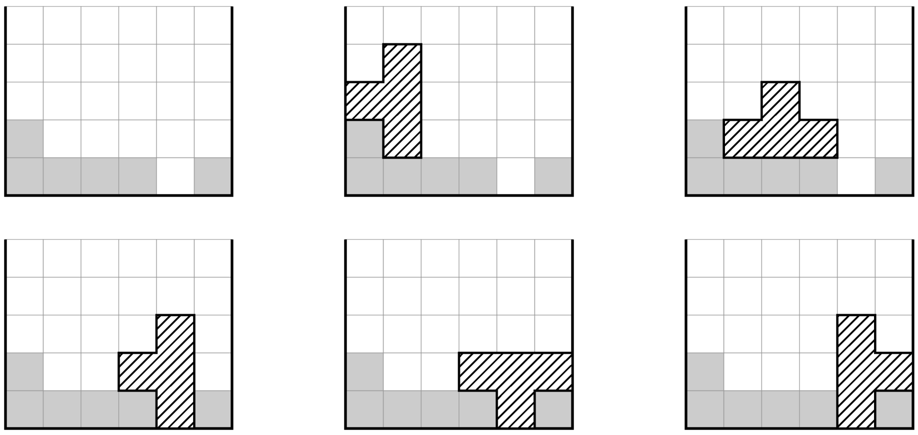
```
1
```

Tetris is a popular computer game played in a field consisting of C columns and an unlimited number of rows. In one move, one of the following seven pieces is dropped into the field:



When dropping the piece, the player is free to rotate the piece 90, 180 or 270 degrees and to move it left or right, as long as the piece stays entirely in the field. The piece then falls until it settles on the bottom of the field or on already occupied squares. In our variant of Tetris the piece **must** fall so that **all parts** of the piece are on the bottom of the field or on already occupied squares. In other words, after the piece falls there may not be **a free square** such that **some square above it is occupied**.

For example, let the field be six columns wide with initial heights (the number of already occupied squares in each column) 2, 1, 1, 1, 0 and 1. Piece number 5 can then be dropped into the field in five different ways:



You are given the initial heights of all columns and the figure to be dropped into the field.

Write a program that calculates the number of different ways to do this, i.e. the number of different field configurations that can be achieved by dropping the piece.

Input

The first line contains two integers C and P, 1 ≤ C ≤ 100, 1 ≤ P ≤ 7, the number of columns and the number of the piece to be dropped.

The second line contains C integers separated by single spaces, each between 0 and 100, inclusive. These are the initial heights of the columns.

Output

Output on a single line the number of different ways to drop the piece in the field.

Sample test data

input

```
6 5
2 1 1 1 0 1
```

output

```
5
```

input

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

output

```
7
```

input

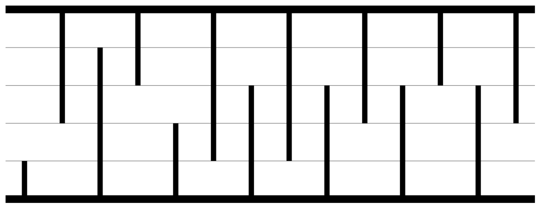
```
9 4
4 3 5 4 6 5 7 6 6
```

output

```
1
```

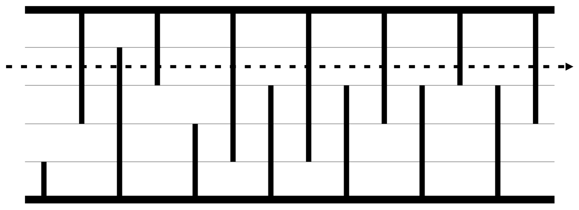
A Japanese firefly has flown into a cave full of obstacles: stalagmites (rising from the floor) and stalactites (hanging from the ceiling). The cave is N units long (where N is even) and H units high. The first obstacle is a stalagmite after which stalactites and stalagmites alternate.

Here is an example cave 14 units long and 5 units high (the image corresponds to the second example):



The Japanese firefly is not the type that would fly around the obstacle, instead it will choose a single height and ram from one end of the cave to the other **destroying all obstacles** in its path with its mighty kung-fu moves.

In the previous example, choosing the 4th level from the ground has the firefly destroying eight obstacles:



This is not the best choice because the firefly will end up less tired if it chooses either level one or five, as that would require destroying only seven obstacles.

You are given the width and length of the cave and the sizes of all obstacles.

Write a program that determines the **minimum number** of obstacles the firefly needs to destroy to reach the end of the cave, and on how many distinct levels it can achieve that number.

Input

The first line contains two integers N and H, 2 ≤ N ≤ 200 000, 2 ≤ H ≤ 500 000, the length and height of the cave. N will always be even.

The next N lines contain the sizes of the obstacles, one per line. All sizes are positive integers less than H.

Output

Output two integers separated by a single space on a single line. The first number is the minimum number of obstacles the firefly has to destroy and the second is the number of levels on which that can be achieved.

Sample test data

input

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

output

```
2 3
```

input

```
14 5
1
3
4
2
2
4
3
4
3
3
3
2
3
3
```

output

```
7 2
```

CIRCLE

One nice summer day while Mirko was drinking lemonade in his room...

"Big brother!", yells Stanko.

"I wonder sometimes which of the two of us is the big one. What is it?", Mirko asked.

"Listen carefully! In the backyard I have N pebbles arranged in a circle. Some of the pebbles are black, some are white. I will do the following: between any two neighbouring pebbles of the **same colour** I will insert a **black pebble**, and between any two neighbouring pebbles of **different colours** I will insert a **white pebble**. At that point there will be 2N pebbles in the circle, so I will remove the starting N pebbles so that only the newly added N pebbles remain. And all this I intend to do **exactly K times**. And then you are to determine my starting circle.", said Stanko long-windedly.

"Ha! I shall not fall prey to your trickery! I can see that it is not necessarily possible to know exactly what the starting circle was, but I can count the number of distinct starting circles that give the same result as your circle after exactly K of those weird transformations of yours", answered Mirko.

You are given the configuration of the circle **before** Stanko performed the transformation described above K times.

Write a program that determines the number of distinct starting circles that give the same circle after K transformations as Stanko's original circle does after K transformations.

Two configurations of pebbles are **considered to be the same circle** if one can be gotten from the other by rotating it any number of positions. For example BBW and BWB is the same circle whereas BBWWBW and WWBBWB are not.

Input

The first line of input contains two integers N and K, $3 \le N \le 100$, $1 \le K \le 10$, where N is the number of pebbles in the circle and K is the number of transformations made by Stanko.

The second line contains exactly N characters 'B' or 'W' representing Stanko's original circle.

Output

Output the number of possible distinct starting circles on a single line.

Sample test data

input

```
3 1
BBW
```

output

```
2
```

input

```
6 2
WBWWBW
```

output

```
3
```

Test case 1 clarification: The circles BBW and WBW become circle BWW after one transformation.