A. Salem and Sticks

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

Salem gave you n sticks with integer positive lengths $a_1, a_2, \ldots, a_n$.

For every stick, you can change its length to any other positive integer length (that is, either shrink or stretch it). The cost of changing the stick's length from a to b is $|a - b|$, where $|x|$ means the absolute value of x.

A stick length a_i is called *almost good* for some integer t if $|a_i - t| \le 1$.

Salem asks you to change the lengths of some sticks (possibly all or none), such that all sticks' lengths are almost good for some positive integer t and the total cost of changing is minimum possible. The value of t is not fixed in advance and you can choose it as any positive integer.

As an answer, print the value of t and the minimum cost. If there are multiple optimal choices for t, print any of them.

Input

The first line contains a single integer n ($1 \le n \le 1000$) — the number of sticks.

The second line contains n integers a_i ($1 \le a_i \le 100$) — the lengths of the sticks.

Output

Print the value of t and the minimum possible cost. If there are multiple optimal choices for t, print any of them.

Examples

input

```
3
10 1 4
```

output

```
3 7
```

input

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

output

```
2 0
```

Note

In the first example, we can change 1 into 2 and 10 into 4 with cost $|1 - 2| + |10 - 4| = 1 + 6 = 7$ and the resulting lengths $[2, 4, 4]$ are almost good for $t = 3$.

In the second example, the sticks lengths are already almost good for $t = 2$, so we don't have to do anything.

B. Zuhair and Strings

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

Given a string s of length n and integer k ($1 \le k \le n$). The string s has a level x, if x is largest non-negative integer, such that it's possible to find in s:

- x **non-intersecting** (non-overlapping) substrings of length k,
- all characters of these x substrings are the same (i.e. each substring contains only one distinct character and this character is the same for all the substrings).

A substring is a sequence of consecutive (adjacent) characters, it is defined by two integers i and j ($1 \le i \le j \le n$), denoted as $s[i \ldots j]$ = "$s_i s_{i+1} \ldots s_j$".

For example, if $k = 2$, then:

- the string "aabb" has level 1 (you can select substring "aa"),
- the strings "zzzz" and "zzbzz" has level 2 (you can select two non-intersecting substrings "zz" in each of them),
- the strings "abed" and "aca" have level 0 (you can't find at least one substring of the length $k = 2$ containing the only distinct character).

Zuhair gave you the integer k and the string s of length n. You need to find x, the level of the string s.

Input

The first line contains two integers n and k ($1 \le k \le n \le 2 \cdot 10^5$) — the length of the string and the value of k.

The second line contains the string s of length n consisting only of lowercase Latin letters.

Output

Print a single integer x — the level of the string.

Examples

input

```
8 2
aaacaabb
```

output

```
2
```

input

```
2 1
ab
```

output

```
1
```

input

```
4 2
abab
```

output

```
0
```

Note

In the first example, we can select 2 non-intersecting substrings consisting of letter 'a': "(aa)ac(aa)bb", so the level is 2.

In the second example, we can select either substring "a" or "b" to get the answer 1.

C. Ayoub and Lost Array

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

Ayoub had an array a of integers of size n and this array had two interesting properties:

- All the integers in the array were between l and r (inclusive).
- The sum of all the elements was divisible by 3.

Unfortunately, Ayoub has lost his array, but he remembers the size of the array n and the numbers l and r, so he asked you to find the number of ways to restore the array.

Since the answer could be very large, print it modulo $10^9 + 7$ (i.e. the remainder when dividing by $10^9 + 7$). In case there are no satisfying arrays (Ayoub has a wrong memory), print 0.

Input

The first and only line contains three integers n, l and r ($1 \le n \le 2 \cdot 10^5, 1 \le l \le r \le 10^9$) — the size of the lost array and the range of numbers in the array.

Output

Print the remainder when dividing by $10^9 + 7$ the number of ways to restore the array.

Examples

input

```
2 1 3
```

output

```
3
```

input

```
3 2 2
```

output

```
1
```

input

```
9 9 99
```

output

```
711426616
```

Note

In the first example, the possible arrays are : $[1, 2], [2, 1], [3, 3]$.

In the second example, the only possible array is $[2, 2, 2]$.

D. Kilani and the Game

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

Kilani is playing a game with his friends. This game can be represented as a grid of size $n \times m$, where each cell is either empty or blocked, and every player has one or more castles in some cells (there are no two castles in one cell).

The game is played in rounds. In each round players expand turn by turn: firstly, the first player expands, then the second player expands and so on. The expansion happens as follows: for each castle the player owns now, he tries to expand into the empty cells nearby. The player i can expand from a cell with his castle to the empty cell if it's possible to reach it in at most s_i (where s_i is player's expansion speed) moves to the left, up, right or down without going through blocked cells or cells occupied by some other player's castle. The player examines the set of cells he can expand to and builds a castle in each of them at once. The turned is passed to the next player after that.

The game ends when no player can make a move. You are given the game field and speed of the expansion for each player. Kilani wants to know for each player how many cells he will control (have a castle their) after the game ends.

Input

The first line contains three integers n, m and p ($1 \le n, m \le 1000$, $1 \le p \le 9$) — the size of the grid and the number of players.

The second line contains p integers s_i ($1 \le s \le 10^9$) — the speed of the expansion for every player.

The following n lines describe the game grid. Each of them consists of m symbols, where '.' denotes an empty cell, '#' denotes a blocked cell and digit x ($1 \le x \le p$) denotes the castle owned by player x.

It is guaranteed, that each player has at least one castle on the grid.

Output

Print p integers — the number of cells controlled by each player after the game ends.

Examples

input

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

output

```
6 3
```

input

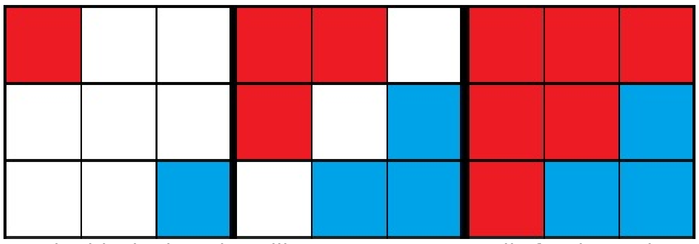
```
3 4 4
1 1 1 1
....
#...
1234
```

output

```
1 4 3 3
```

Note

The picture below show the game before it started, the game after the first round and game after the second round in the first example:



In the second example, the first player is "blocked" so he will not capture new cells for the entire game. All other player will expand up during the first two rounds and in the third round only the second player will move to the left.

E. Helping Hiasat

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

Hiasat registered a new account in NeckoForces and when his friends found out about that, each one of them asked to use his name as Hiasat's handle.

Luckily for Hiasat, he can change his handle in some points in time. Also he knows the exact moments friends will visit his profile page. Formally, you are given a sequence of events of two types:

- 1 — Hiasat can change his handle.
- 2 s — friend s visits Hiasat's profile.

The friend s will be happy, if each time he visits Hiasat's profile his handle would be s.

Hiasat asks you to help him, find the maximum possible number of happy friends he can get.

Input

The first line contains two integers n and m $(1 \le n \le 10^5, 1 \le m \le 40)$ — the number of events and the number of friends.

Then n lines follow, each denoting an event of one of two types:

- 1 — Hiasat can change his handle.
- 2 s — friend s $(1 \le |s| \le 40)$ visits Hiasat's profile.

It's guaranteed, that each friend's name consists only of lowercase Latin letters.

It's guaranteed, that the first event is always of the first type and each friend will visit Hiasat's profile at least once.

Output

Print a single integer — the maximum number of happy friends.

Examples

input

```
5 3
1
2 motarack
2 mike
1
2 light
```

output

```
2
```

input

```
4 3
1
2 alice
2 bob
2 tanyaromanova
```

output

```
1
```

Note

In the first example, the best way is to change the handle to the "motarack" in the first event and to the "light" in the fourth event. This way, "motarack" and "light" will be happy, but "mike" will not.

In the second example, you can choose either "alice", "bob" or "tanyaromanova" and only that friend will be happy.