Problem A. Cave

Input file:	`standard input`
Output file:	`standard output`
Time limit (C++):	3 seconds
Time limit (Java):	6 seconds
Time limit (Python):	30 seconds
Memory limit:	128 megabytes
Java Class Name:	`cave`
Maximum Points Available:	100

For reasons unknown, Bruce finds himself waking up in a large cave. Fortunately, he seems to have a map of the layout of the cave, as well as K sticks of dynamite to blast his way to the exit. As Bruce is afraid of the dark, he wishes to escape as quickly as possible, and has thus asked you to write a program which will determine his shortest escape route.

The cave is modeled as an N by M grid of cells. Each cell is either solid or empty. A valid escape route consists of Bruce taking a number of steps from his starting location to finish at the exit. In each step, Bruce can move from his current cell to an adjacent empty cell. If he has at least one stick of dynamite left, he may instead use a stick of dynamite to move to an adjacent solid cell. The length of the escape route is the number of steps it consists of.

Input

The first line contains three space-separated integers N ($1 \leq N \leq 500$), M ($1 \leq M \leq 500$), and K ($1 \leq K \leq 50$).

The following N lines each contain M characters, depicting the cave. '`.`' denotes an empty cell. '`#`' denotes a solid cell. '`S`' denotes the empty cell where Bruce starts, and occurs exactly once in the input. '`E`' denotes the empty cell where Bruce can exit the cave, and occurs exactly once in the input.

Output

A single integer, which is the length of the shortest valid escape route. If there is no such escape route, output `-1` instead.

Scoring

Subtask 1 (10 points): $N = 1$

Subtask 2 (10 points): There is no cell of type '`#`'.

Subtask 3 (25 points): $K = 0$

Subtask 4 (20 points): $K = 1$

Subtask 5 (35 points): No further restrictions

Example

standard input	standard output
6 8 2 .S...... ######## .###.... .#.##### .#...### ...#..#E	15

Problem B. Tigers

Input file:	`standard input`
Output file:	`standard output`
Time limit (C++):	2 seconds
Time limit (Java):	4 seconds
Time limit (Python):	20 seconds
Memory limit:	64 megabytes
Java Class Name:	`tigers`
Maximum Points Available:	100

Alexandra was walking through the Siberian forest one day when she suddenly noticed she was being watched. Up atop a cliff face in front of her was Boris the Siberian Tiger, lingering ominously. Alexandra is best friends with Boris, but she knows that Boris might decide to eat her if he feels hungry. Alexandra would therefore like to work out how many ways Boris can jump down the cliff face to the ground to eat her so she can plan her escape if necessary.

The cliff consists of N ledges. Ledge i is h_i units above the ground and is x_i units to the right of Alexandra (and can be considered a single point on the cliff face). Boris starts at the highest ledge, and repeatedly jumps down to a ledge strictly lower (or directly to the ground) until he reaches the ground. Specifically, if Boris is on ledge i, he may jump directly to the ground, or to any ledge j which has height $h_j < h_i$ and $|x_j - x_i| \leq L$, where L is Boris's jumping range.

Alexandra would like to know the number of ways Boris can jump between ledges to reach the ground. Since the answer to Alexandra's question may be large, you should output the the remainder of the answer when divided by $10^9 + 7$. Help Alexandra avoid being eaten by Boris.

Input

The first line consists of two space-separated integers, N ($1 \leq N \leq 100\,000$) and L ($0 \leq L \leq 100\,000$).

The next N lines contain two integers each. On the ith of these lines is x_i ($0 \leq x_i \leq 100\,000$) and h_i ($1 \leq h_i \leq 1\,000\,000$), describing the position of the ith ledge (where $1 \leq i \leq N$).

It is guaranteed that the heights of the ledges will be distinct.

Output

Output a single integer denoting the number of ways Boris can jump between ledges to reach the ground, modulo $10^9 + 7$.

Scoring

Subtask 1 (10 Points): $N \leq 6$

Subtask 2 (15 Points): $N \leq 100\,000, L = 100\,000$

Subtask 3 (35 Points): $N \leq 1\,000$

Subtask 4 (40 Points): $N \leq 100\,000$

Example

standard input	standard output
3 2 5 10 7 8 8 5	3

Note

In the sample input, there are ledges at points $(5, 10)$, $(7, 8)$ and $(8, 5)$ respectively. Boris starts at the highest ledge $(5, 10)$. Here are all the possible ways Boris can reach the ground:

1. $(5, 10) \longrightarrow$ ground
2. $(5, 10) \longrightarrow (7, 8) \longrightarrow$ ground
3. $(5, 10) \longrightarrow (7, 8) \longrightarrow (8, 5) \longrightarrow$ ground

Problem C. Mastermind

Time limit (C++):	5 seconds
Time limit (Java):	10 seconds
Time limit (Python):	50 seconds
Memory limit:	1024 megabytes
Java Class Name:	`mastermind`
Maximum Points Available:	100

Mastermind is a popular two-player game first invented in the 1970s. One player is the *codemaker* and the other player is the *codebreaker*. The ultimate goal of the game is for the *codebreaker* to guess a secret pattern that the *codemaker* chooses at the start.

The game starts by having the *codemaker* think of a secret pattern of length N, using positive integers ranging from 1 to K. Duplicate numbers are allowed. The *codebreaker* then attempts to guess the pattern the *codemaker* has thought of. The *codebreaker* presents his guess to the *codemaker*, after which the *codemaker* provides feedback by giving 2 numbers A, B. The first number A denotes the number of **correct** colours in the **correct** positions, and the second number B denotes the number of **correct** colours in the **wrong** positions.

After the codebreaker received his feedback, he should make another guess, after which more feedback will be given. This continues back and forth until the codebreaker correctly guesses the pattern the codemaker thought of.

Note: If there are duplicate numbers in the guess, they cannot all be counted unless they correspond to the same number of duplicate numbers in the secret pattern. For example, if the secret pattern is `4 1 1 1` and the codebreaker guesses `2 4 4 3`, then the judge will return `0 1`. Similarly, if the guess `4 4 3 3` is made, the judge will return `1 0`.

Formally speaking, if P_i denotes the number of times i appears in the secret pattern and G_i denotes the number of times i appears in the guess, then B can be calculated as:

$$B = \left(\sum_{i=1}^{K} \min(P_i, G_i)\right) - A = (\min(P_1, G_1) + \min(P_2, G_2) + \cdots + \min(P_K, G_K)) - A$$

Note: There is no intrinsic limit to the number of guesses you are allowed. Rather your score for a particular task will be calculated based on the number of guesses you needed before correctly guessing the codemaker's pattern.

Interaction Protocol

This is an interactive task. You are thus required to compile your code against a stub that will perform the interaction with the grader for you. You are required to write two functions, `init` and `makeGuess`. The functions must have the following signatures:

Python
```
def init(N,K)
def makeGuess(A, B)
```

C++
```
vector<int> init(int N, int K);
vector<int> makeGuess(int A, int B)
```

Java
```
public static int[] init(int N, int M)
public static int[] makeGuess(int A, int B)
```

When your program is run, the function `init` will be called once with the parameters N, K. The function `init` should then output a list/array of integers containing an initial guess of the secret pattern.

Subsequently, the stub will call then `makeMove` with the values of A and B until you correctly guess the secret pattern. For every function call `makeMove`, you must return another guess for the pattern.

Scoring

Subtask 1 (at most 10 points): $N \leq 3, K \leq 2$

Subtask 2 (at most 16 points): $N \leq 2$

Subtask 3 (at most 16 points): $K \leq 2$

Subtask 4 (at most 18 points): $N, K \leq 5$

Subtask 5 (at most 20 points): $N, K \leq 8$

Subtask 6 (at most 20 points): No further constraints

Each test case can give a maximum of 2 points. If your program correctly guesses the secret pattern in G guesses, then your score S for that particular test case is:

$$S = \min\left(\frac{4(N+K)}{3G}, 2\right) \text{ points}$$

Example

Function Call	Function Response
`init(4,6)`	`{2, 5, 3, 6}`
`makeGuess(1,1)`	`{2, 4, 5, 6}`
`makeGuess(1,1)`	`{4, 6, 3, 1}`
`makeGuess(2,0)`	`{4, 2, 3, 6}`
`makeGuess(2,1)`	`{4, 4, 3, 2}`

In the given example, the codemaker creates the secret pattern: `4 4 3 2`. The codebreaker then begins by guessing the sequence `2 5 3 6`. He has one number, 3, in the correct position and another number, 2, which is correct, but in the wrong position, hence the judge gives a result of `1 1`.

He guesses again, this time getting 4 in the correct position and still 2 in the wrong position, hence the judge returns `1 1` again.

His third guess gets two numbers in the correct position: the numbers 4 and 3. No other number is correct, so the judge returns `2 0`

His fourth guess gets both 4 and 3 in the correct position and 2 in the wrong position, so the judge returns `2 1`

His final guess matches the codemaker's secret pattern exactly. The judge therefore exits and the pattern was correctly guessed in $G = 5$ guesses.

Note

Testing your program is made easier with the supplied stub code and manager.

To compile and test your program, run the script `run.sh` with the name of an input file and the name of your source code. The script will automatically compile your code against the stub, and run it against the grader. It will output your code's responses to the input.

The format of the input file is as follows. The first line must contain two space separated integers, N and K. The next line contains N space-separated integers denoting the secret pattern to guess.

For example, to test a solution written in java against the sample input, you must run

```
./run.sh mastermind.java sample.in
```

If your stubs, graders or compilation scripts are corrupted during the contest, you may re-download them from the problem statement on the web interface.