A. Satisfying Constraints

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Alex is solving a problem. He has n constraints on what the integer k can be. There are three types of constraints:

1. k must be **greater than or equal to** some integer x;
2. k must be **less than or equal to** some integer x;
3. k must be **not equal to** some integer x.

Help Alex find the number of integers k that satisfy all n constraints. It is guaranteed that the **answer is finite** (there exists at least one constraint of type 1 and at least one constraint of type 2). Also, it is guaranteed that **no two constraints are the exact same**.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 500$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 100$) — the number of constraints.

The following n lines describe the constraints. Each line contains two integers a and x ($a \in \{1, 2, 3\}$, $1 \le x \le 10^9$). a denotes the type of constraint. If $a = 1$, k must be greater than or equal to x. If $a = 2$, k must be less than or equal to x. If $a = 3$, k must be not equal to x.

It is guaranteed that there is a finite amount of integers satisfying all n constraints (there exists at least one constraint of type 1 and at least one constraint of type 2). It is also guaranteed that no two constraints are the exact same (in other words, all pairs (a, x) are distinct).

Output

For each test case, output a single integer — the number of integers k that satisfy all n constraints.

Standard Input	Standard Output
6 4 1 3 2 10 3 1 3 5 2 1 5 2 4 10 3 6 3 7 1 2 1 7 3 100 3 44	7 0 90 0 0 800000000

2 100 2 98 1 3 3 99 6 1 5 2 10 1 9 2 2 3 2 3 9 5 1 1 2 2 3 1 3 2 3 3 6 1 10000 2 900000000 3 500000000 1 100000000 3 10000 3 900000001	

Note

In the first test case, $k \geq 3$ and $k \leq 10$. Furthermore, $k \neq 1$ and $k \neq 5$. The possible integers k that satisfy the constraints are $3, 4, 6, 7, 8, 9, 10$. So the answer is 7.

In the second test case, $k \geq 5$ and $k \leq 4$, which is impossible. So the answer is 0.

,

B. Summation Game

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Alice and Bob are playing a game. They have an array $a_1, a_2, \ldots, a_n$. The game consists of two steps:

- First, Alice will remove **at most** k elements from the array.
- Second, Bob will multiply **at most** x elements of the array by -1.

Alice wants to maximize the sum of elements of the array while Bob wants to minimize it. Find the sum of elements of the array after the game if both players play optimally.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains three integers n, k, and x ($1 \le n \le 2 \cdot 10^5$, $1 \le x, k \le n$) — the number of elements in the array, the limit on the number of elements of the array that Alice can remove, and the limit on the number of elements of the array that Bob can multiply -1 to.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 1000$) — the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the sum of elements of the array after the game if both players play optimally.

Standard Input	Standard Output
8 1 1 1 1 4 1 1 3 1 2 4 6 6 3 1 4 3 2 5 6 6 6 1 3 7 3 3 32 15 8 5 3 5 5 3 3 3 2 9 9 10 6 4 1 8 2 9 3 3 4 5 3 200 2 2 1 4 3 2 1 2 1 3	0 2 0 3 -5 -9 0 -1

Note

In the first test case, it is optimal for Alice to remove the only element of the array. Then, the sum of elements of the array is 0 after the game is over.

In the second test case, it is optimal for Alice to not remove any elements. Bob will then multiply 4 by -1. So the final sum of elements of the array is $3 + 1 + 2 - 4 = 2$.

In the fifth test case, it is optimal for Alice to remove $9, 9$. Bob will then multiply $5, 5, 3$ by -1. So the final sum of elements of the array is $-5 - 5 - 3 + 3 + 3 + 2 = -5$.

,

C. Partitioning the Array

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

Allen has an array $a_1, a_2, \ldots, a_n$. For every positive integer k that is a divisor of n, Allen does the following:

- He partitions the array into $\frac{n}{k}$ disjoint subarrays of length k. In other words, he partitions the array into the following subarrays:

$$[a_1, a_2, \ldots, a_k], [a_{k+1}, a_{k+2}, \ldots, a_{2k}], \ldots, [a_{n-k+1}, a_{n-k+2}, \ldots, a_n]$$

- Allen earns one point if there exists some positive integer m ($m \geq 2$) such that if he replaces every element in the array with its remainder when divided by m, then all subarrays will be identical.

Help Allen find the number of points he will earn.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the number of points Allen will earn.

Standard Input	Standard Output
8 4 1 2 1 4 3 1 2 3 5 1 1 1 1 1 6 1 3 1 1 3 1 6 6 2 6 2 2 2 6 2 6 3 6 6 6 10 1 7 5 1 4 3 1 3 1 4 1 1	2 1 2 4 4 1 2 1

Note

In the first test case, $k = 2$ earns a point since Allen can pick $m = 2$ and both subarrays will be equal to $[1, 0]$. $k = 4$ also earns a point, since no matter what m Allen chooses, there will be only one subarray and thus all subarrays are equal.

In the second test case, Allen earns 1 point for $k = 3$, where his choice for m does not matter.

,

D. Array Repetition

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 256 megabytes

Jayden has an array a which is initially empty. There are n operations of two types he must perform in the given order.

1. Jayden appends an integer x ($1 \le x \le n$) to the end of array a.
2. Jayden appends x copies of array a to the end of array a. In other words, array a becomes $[a, \underbrace{a, \ldots, a}_{x}]$. It is guaranteed that he has done at least one operation of the first type before this.

Jayden has q queries. For each query, you must tell him the k-th element of array a. The elements of the array are numbered from 1.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 5000$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and q ($1 \le n, q \le 10^5$) — the number of operations and the number of queries.

The following n lines describe the operations. Each line contains two integers b and x ($b \in \{1, 2\}$), where b denotes the type of operation. If $b = 1$, then x ($1 \le x \le n$) is the integer Jayden appends to the end of the array. If $b = 2$, then x ($1 \le x \le 10^9$) is the number of copies Jayden appends to the end of the array.

The next line of each test case contains q integers $k_1, k_2, \ldots, k_q$ ($1 \le k_i \le \min(10^{18}, c)$), which denote the queries, where c is the size of the array after finishing all n operations.

It is guaranteed that the sum of n and the sum of q over all test cases does not exceed 10^5.

Output

For each test case, output q integers — answers to Jayden's queries.

Standard Input	Standard Output
4 5 10 1 1 1 2 2 1 1 3 2 3 1 2 3 4 5 6 14 15 16 20 10 10 1 3 1 8 2 15 1 6	1 2 1 2 3 1 2 3 1 3 9 8 1 3 1 3 6 3 8 8 11 11 11 10 11 1 2

1 9 1 1 2 6 1 1 2 12 2 10 32752 25178 3198 3199 2460 2461 31450 33260 9016 4996 12 5 1 6 1 11 2 392130334 1 4 2 744811750 1 10 1 5 2 209373780 2 178928984 1 3 2 658326464 2 1000000000 914576963034536490 640707385283752918 636773368365261971 584126563607944922 1000000000000000000 2 2 1 1 1 2 1 2	

Note

In the first test case:

- After the first operation $a = [1]$;
- After the second operation $a = [1, 2]$;
- After the third operation $a = [1, 2, 1, 2]$;
- After the fourth operation $a = [1, 2, 1, 2, 3]$;
- After the fifth operation $a = [1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3, 1, 2, 1, 2, 3]$.

In the fourth test case, after all operations $a = [1, 2]$.

,

E. Counting Binary Strings

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

Patrick calls a substring† of a binary string‡ *good* if this substring contains exactly one 1.

Help Patrick count the number of binary strings s such that s contains exactly n good substrings and has no good substring of length strictly greater than k. Note that substrings are differentiated by their location in the string, so if $s = 1010$ you should count both occurrences of 10.

† A string a is a substring of a string b if a can be obtained from b by the deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

‡ A binary string is a string that only contains the characters 0 and 1.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 2500$) — the number of test cases. The description of the test cases follows.

The only line of each test case contains two integers n and k ($1 \le n \le 2500$, $1 \le k \le n$) — the number of required good substrings and the maximum allowed length of a good substring.

It is guaranteed that the sum of n over all test cases does not exceed 2500.

Output

For each test case, output a single integer — the number of binary strings s such that s contains exactly n good substrings and has no good substring of length strictly greater than k. Since this integer can be too large, output it modulo $998\,244\,353$.

Standard Input	Standard Output
6 1 1 3 2 4 2 5 4 6 2 2450 2391	1 3 5 12 9 259280854

Note

In the first test case, the only suitable binary string is 1. String 01 is not suitable because it contains a substring 01 with length $2 > 1$.

In the second test case, suitable binary strings are 011, 110 and 111.

In the third test case, suitable binary strings are 101, 0110, 0111, 1110, and 1111.

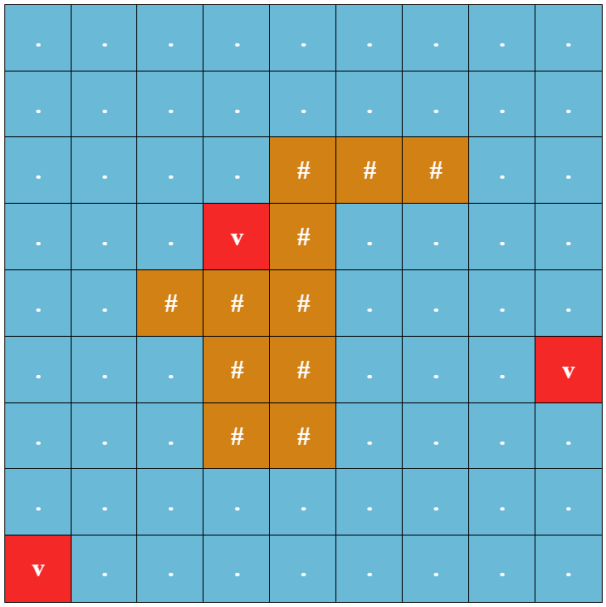
,

F1. Smooth Sailing (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     5 seconds
Memory limit:   1024 megabytes
```

The only difference between the two versions of this problem is the constraint on q. You can make hacks only if both versions of the problem are solved.

Thomas is sailing around an island surrounded by the ocean. The ocean and island can be represented by a grid with n rows and m columns. The rows are numbered from 1 to n from top to bottom, and the columns are numbered from 1 to m from left to right. The position of a cell at row r and column c can be represented as (r, c). Below is an example of a valid grid.

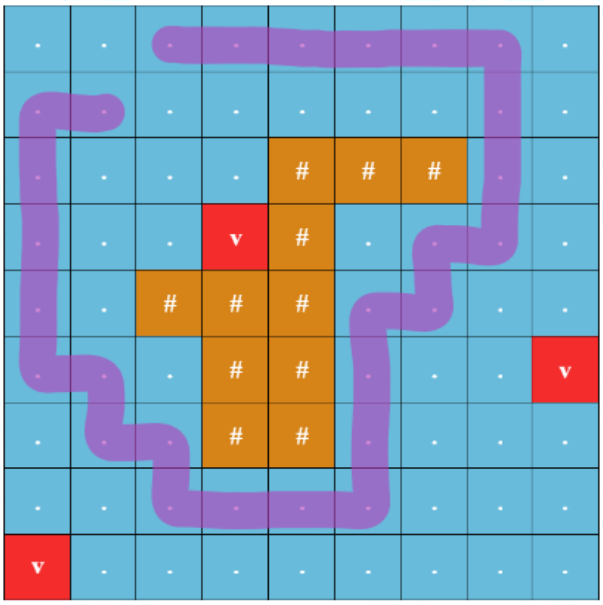


Example of a valid grid

There are three types of cells: island, ocean and underwater volcano. Cells representing the island are marked with a '#', cells representing the ocean are marked with a '.', and cells representing an underwater volcano are marked with a 'v'. It is guaranteed that there is at least one island cell and at least one underwater volcano cell. It is also guaranteed that the set of all island cells forms a single connected component[†] and the set of all ocean cells and underwater volcano cells forms a single connected component. Additionally, it is guaranteed that there are no island cells at the edge of the grid (that is, at row 1, at row n, at column 1, and at column m).

Define a *round trip* starting from cell (x, y) as a path Thomas takes which satisfies the following conditions:

- The path starts and ends at (x, y).
- If Thomas is at cell (i, j), he can go to cells $(i + 1, j)$, $(i - 1, j)$, $(i, j - 1)$, and $(i, j + 1)$ as long as the destination cell **is an ocean cell or an underwater volcano cell** and is still inside the grid. Note that it is allowed for Thomas to visit the same cell multiple times in the same round trip.
- The path must go around the island and fully encircle it. Some path p fully encircles the island if it is impossible to go from an island cell to a cell on the grid border by only traveling **to adjacent on a side or diagonal** cells without visiting a cell on path p. In the image below, the path starting from $(2, 2)$, going to $(1, 3)$, and going back to $(2, 2)$ the other way does **not** fully encircle the island and is not considered a round trip.



Example of a path that does **not** fully encircle the island

The *safety* of a round trip is the minimum Manhattan distance[‡] from a cell on the round trip to an underwater volcano (note that the presence of island cells does not impact this distance).

You have q queries. A query can be represented as (x, y) and for every query, you want to find the maximum safety of a round trip starting from (x, y). It is guaranteed that (x, y) is an ocean cell or an underwater volcano cell.

[†] A set of cells forms a single connected component if from any cell of this set it is possible to reach any other cell of this set by moving only through the cells of this set, each time going to a cell **with a common side**.

[‡] Manhattan distance between cells (r_1, c_1) and (r_2, c_2) is equal to $|r_1 - r_2| + |c_1 - c_2|$.

Input

The first line contains three integers n, m, and q ($3 \le n, m \le 10^5$, $9 \le n \cdot m \le 3 \cdot 10^5$, $1 \le q \le 5$) — the number of rows and columns of the grid and the number of queries.

Each of the following n lines contains m characters describing the cells of the grid. The character '#' denotes an island cell, '.' denotes an ocean cell, and 'V' denotes an underwater volcano cell.

It is guaranteed that there is at least one island cell and at least one underwater volcano cell. It is guaranteed that the set of all island cells forms a single connected component and the set of all ocean cells and underwater volcano cells forms a single connected component. Also, it is guaranteed that there are no island cells at the edge of the grid (that is, at the row 1, at the row n, at the column 1, and at the column m).

The following q lines describe the queries. Each of these lines contains two integers x and y ($1 \le x \le n$, $1 \le y \le m$) denoting a round trip starting from (x, y).

It is guaranteed that (x, y) is an ocean cell or an underwater volcano cell.

Output

For each query, output a single integer — the maximum safety of a round trip starting from the specified position.

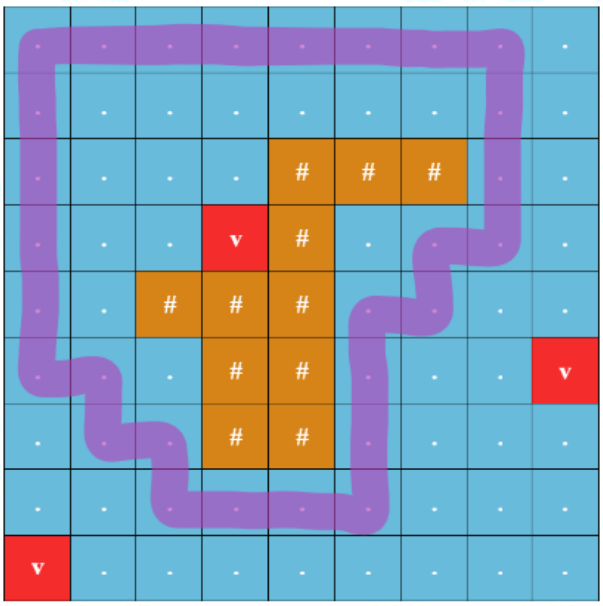
Standard Input	Standard Output
9 9 3	3 0 3

....###.. ...v#.... ..###.... ...##...v ...##.... .... v........ 1 1 9 1 5 7	
3 3 5 ..v .#. ... 1 2 1 3 2 3 2 1 3 2	0 0 0 0 0
14 13 5 ....vvvvvvv... ...v.....v... ...v.###.v... ...v.#.#.v... ...v..v..v... ...v..v..v... ....v...v.... .....vvv..... .... 1 1 7 7 5 6 4 10 13 6	3 0 1 0 2
10 11 4 ....#######.. ..#..#..#.. ..#.....#.. ..#..v..#.. ..#.###.#.. ..#.#.#.#..	1 2 3 4

..#...#.#.. ..#####.#.. .... 7 6 3 7 6 8 1 1	

Note

For the first example, the image below shows an optimal round trip starting from $(1, 1)$. The round trip has a safety of 3 as the minimum Manhattan distance from a cell on the round trip to an underwater volcano is 3.



Example of an optimal round trip

For the fourth example, remember that it is allowed for Thomas to visit the same cell multiple times in the same round trip. For example, doing so is necessary for the round trip starting from $(7, 6)$.

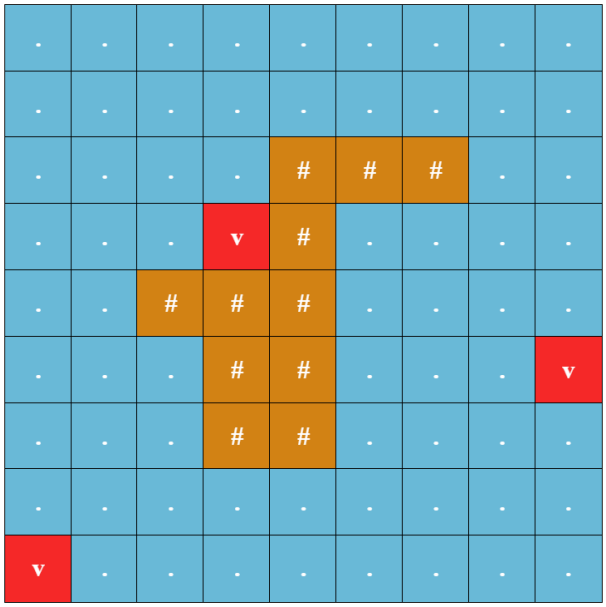
,

F2. Smooth Sailing (Hard Version)

Input file: standard input
Output file: standard output
Time limit: 5 seconds
Memory limit: 1024 megabytes

The only difference between the two versions of this problem is the constraint on q. You can make hacks only if both versions of the problem are solved.

Thomas is sailing around an island surrounded by the ocean. The ocean and island can be represented by a grid with n rows and m columns. The rows are numbered from 1 to n from top to bottom, and the columns are numbered from 1 to m from left to right. The position of a cell at row r and column c can be represented as (r, c). Below is an example of a valid grid.

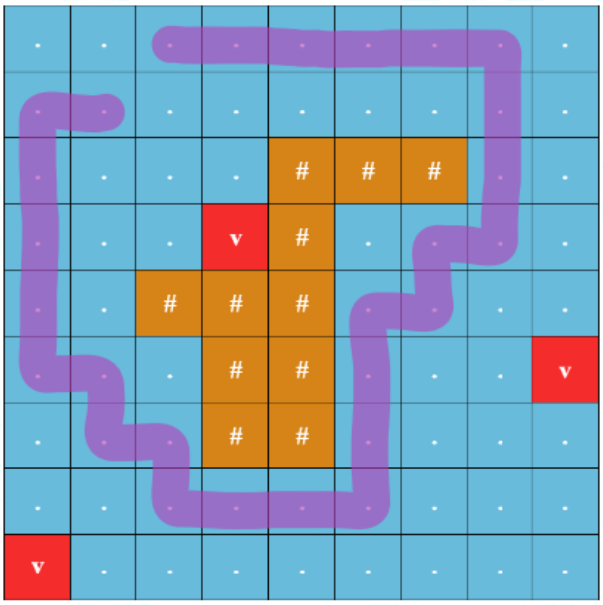


Example of a valid grid

There are three types of cells: island, ocean and underwater volcano. Cells representing the island are marked with a '#', cells representing the ocean are marked with a '.', and cells representing an underwater volcano are marked with a 'v'. It is guaranteed that there is at least one island cell and at least one underwater volcano cell. It is also guaranteed that the set of all island cells forms a single connected component[†] and the set of all ocean cells and underwater volcano cells forms a single connected component. Additionally, it is guaranteed that there are no island cells at the edge of the grid (that is, at row 1, at row n, at column 1, and at column m).

Define a *round trip* starting from cell (x, y) as a path Thomas takes which satisfies the following conditions:

- The path starts and ends at (x, y).
- If Thomas is at cell (i, j), he can go to cells $(i + 1, j)$, $(i - 1, j)$, $(i, j - 1)$, and $(i, j + 1)$ as long as the destination cell **is an ocean cell or an underwater volcano cell** and is still inside the grid. Note that it is allowed for Thomas to visit the same cell multiple times in the same round trip.
- The path must go around the island and fully encircle it. Some path p fully encircles the island if it is impossible to go from an island cell to a cell on the grid border by only traveling **to adjacent on a side or diagonal** cells without visiting a cell on path p. In the image below, the path starting from $(2, 2)$, going to $(1, 3)$, and going back to $(2, 2)$ the other way does **not** fully encircle the island and is not considered a round trip.



Example of a path that does **not** fully encircle the island

The *safety* of a round trip is the minimum Manhattan distance[‡] from a cell on the round trip to an underwater volcano (note that the presence of island cells does not impact this distance).

You have q queries. A query can be represented as (x, y) and for every query, you want to find the maximum safety of a round trip starting from (x, y). It is guaranteed that (x, y) is an ocean cell or an underwater volcano cell.

[†] A set of cells forms a single connected component if from any cell of this set it is possible to reach any other cell of this set by moving only through the cells of this set, each time going to a cell **with a common side**.

[‡] Manhattan distance between cells (r_1, c_1) and (r_2, c_2) is equal to $|r_1 - r_2| + |c_1 - c_2|$.

Input

The first line contains three integers n, m, and q ($3 \le n, m \le 10^5$, $9 \le n \cdot m \le 3 \cdot 10^5$, $1 \le q \le 3 \cdot 10^5$) — the number of rows and columns of the grid and the number of queries.

Each of the following n lines contains m characters describing the cells of the grid. The character '#' denotes an island cell, '.' denotes an ocean cell, and 'V' denotes an underwater volcano cell.

It is guaranteed that there is at least one island cell and at least one underwater volcano cell. It is guaranteed that the set of all island cells forms a single connected component and the set of all ocean cells and underwater volcano cells forms a single connected component. Also, it is guaranteed that there are no island cells at the edge of the grid (that is, at the row 1, at the row n, at the column 1, and at the column m).

The following q lines describe the queries. Each of these lines contains two integers x and y ($1 \le x \le n$, $1 \le y \le m$) denoting a round trip starting from (x, y).

It is guaranteed that (x, y) is an ocean cell or an underwater volcano cell.

Output

For each query, output a single integer — the maximum safety of a round trip starting from the specified position.

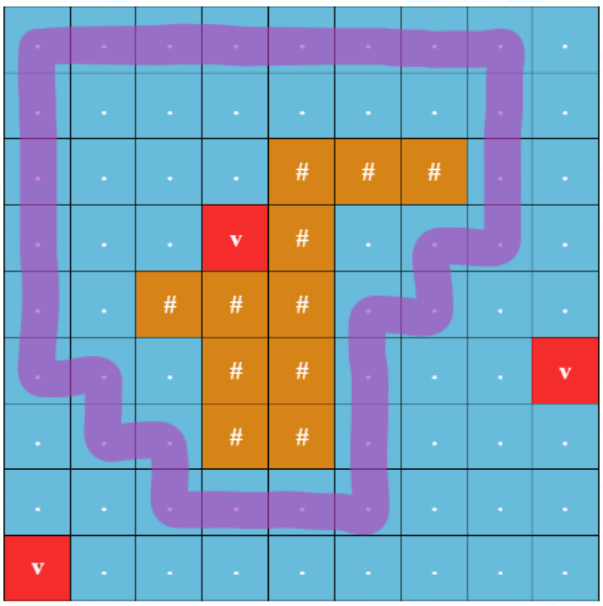
Standard Input	Standard Output
9 9 3	3 0 3

....###.. ...v#.... ..###.... ...##...v ...##.... .... v........ 1 1 9 1 5 7	
3 3 5 ..v .#. ... 1 2 1 3 2 3 2 1 3 2	0 0 0 0 0
14 13 5 ....vvvvvvv... ...v.....v... ...v.###.v... ...v.#.#.v... ...v..v..v... ...v..v..v... ....v...v.... .....vvv..... .... 1 1 7 7 5 6 4 10 13 6	3 0 1 0 2
10 11 4 ....#######.. ..#..#..#.. ..#.....#.. ..#..v..#.. ..#.###.#.. ..#.#.#.#..	1 2 3 4

..#...#.#.. ..#####.#.. .... 7 6 3 7 6 8 1 1	

Note

For the first example, the image below shows an optimal round trip starting from $(1, 1)$. The round trip has a safety of 3 as the minimum Manhattan distance from a cell on the round trip to an underwater volcano is 3.



Example of an optimal round trip

For the fourth example, remember that it is allowed for Thomas to visit the same cell multiple times in the same round trip. For example, doing so is necessary for the round trip starting from $(7, 6)$.