

## Codeforces Round #192 (Div. 1)

### A. Purification

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

You are an adventurer currently journeying inside an evil temple. After defeating a couple of weak zombies, you arrived at a square room consisting of tiles forming an  $n \times n$  grid. The rows are numbered  $1$  through  $n$  from top to bottom, and the columns are numbered  $1$  through  $n$  from left to right. At the far side of the room lies a door locked with evil magical forces. The following inscriptions are written on the door:

*The cleaning of all evil will awaken the door!*

Being a very senior adventurer, you immediately realize what this means. You notice that every single cell in the grid are initially evil. You should purify all of these cells.

The only method of tile purification known to you is by casting the "Purification" spell. You cast this spell on a single tile — then, all cells that are located in the same row and all cells that are located in the same column as the selected tile become purified (including the selected tile)! It is allowed to purify a cell more than once.

You would like to purify all  $n \times n$  cells while minimizing the number of times you cast the "Purification" spell. This sounds very easy, but you just noticed that some tiles are particularly more evil than the other tiles. You cannot cast the "Purification" spell on those particularly more evil tiles, not even after they have been purified. They can still be purified if a cell sharing the same row or the same column gets selected by the "Purification" spell.

Please find some way to purify all the cells with the minimum number of spells cast. Print -1 if there is no such way.

#### Input

The first line will contain a single integer  $n$  ( $1 \leq n \leq 100$ ). Then,  $n$  lines follows, each contains  $n$  characters. The  $j$ -th character in the  $i$ -th row represents the cell located at row  $i$  and column  $j$ . It will be the character 'E' if it is a particularly more evil cell, and '.' otherwise.

#### Output

If there exists no way to purify all the cells, output -1. Otherwise, if your solution casts  $X$  "Purification" spells (where  $X$  is the minimum possible number of spells), output  $X$  lines. Each line should consist of two integers denoting the row and column numbers of the cell on which you should cast the "Purification" spell.

#### Examples

input
3 .E. E.E .E.
output
1 1 2 2 3 3
input
3 EEE E.. E.E
output
-1
input
5 EE.EE E.EE. E...E .EE.E EE.EE
output

3	3
1	3
2	2
4	4
5	3

**Note**

The first example is illustrated as follows. Purple tiles are evil tiles that have not yet been purified. Red tile is the tile on which "Purification" is cast. Yellow tiles are the tiles being purified as a result of the current "Purification" spell. Green tiles are tiles that have been purified previously.



In the second example, it is impossible to purify the cell located at row 1 and column 1.

For the third example:



## B. Biridian Forest

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

You're a mikemon breeder currently in the middle of your journey to become a mikemon master. Your current obstacle is go through the infamous Biridian Forest.

### The forest

The Biridian Forest is a two-dimensional grid consisting of  $r$  rows and  $c$  columns. Each cell in Biridian Forest may contain a tree, or may be vacant. A vacant cell may be occupied by zero or more mikemon breeders (there may also be breeders other than you in the forest). Mikemon breeders (including you) cannot enter cells with trees. One of the cells is designated as the exit cell.

The initial grid, including your initial position, the exit cell, and the initial positions of all other breeders, will be given to you. Here's an example of such grid (from the first example):



### Moves

Breeders (including you) may move in the forest. In a single move, breeders may perform one of the following actions:

- Do nothing.
- Move from the current cell to one of the four adjacent cells (two cells are adjacent if they share a side). Note that breeders cannot enter cells with trees.
- If you are located on the exit cell, you may leave the forest. Only you can perform this move — all other mikemon breeders will never leave the forest by using this type of movement.

After each time you make a single move, each of the other breeders simultaneously make a single move (the choice of which move to make may be different for each of the breeders).

### Mikemon battle

If you and  $t$  ( $t > 0$ ) mikemon breeders are located on the same cell, exactly  $t$  mikemon battles will ensue that time (since you will be battling each of those  $t$  breeders once). After the battle, all of those  $t$  breeders will leave the forest to heal their respective mikemons.

Note that the moment you leave the forest, no more mikemon battles can ensue, even if another mikemon breeder move to the exit cell immediately after that. Also note that a battle only happens between you and another breeders — there will be no battle between two other breeders (there may be multiple breeders coexisting in a single cell).

### Your goal

You would like to leave the forest. In order to do so, you have to make a sequence of moves, ending with a move of the final type. Before you make any move, however, you post this sequence on your personal virtual idol Blog. Then, you will follow this sequence of moves faithfully.

### Goal of other breeders

Because you post the sequence in your Blog, the other breeders will all know your exact sequence of moves even before you make your first move. All of them will move in such way that will guarantee a mikemon battle with you, if possible. The breeders that couldn't battle you will do nothing.

### Your task

Print the minimum number of mikemon battles that you must participate in, assuming that you pick the sequence of moves that minimize this number. Note that you are not required to minimize the number of moves you make.

### Input

The first line consists of two integers:  $r$  and  $c$  ( $1 \leq r, c \leq 1000$ ), denoting the number of rows and the number of columns in Biridian Forest. The next  $r$  rows will each depict a row of the map, where each character represents the content of a single cell:

- 'T': A cell occupied by a tree.
- 'S': An empty cell, and your starting position. There will be exactly one occurrence of this in the map.
- 'E': An empty cell, and where the exit is located. There will be exactly one occurrence of this in the map.
- A digit (0-9): A cell represented by a digit  $X$  means that the cell is empty and is occupied by  $X$  breeders (in particular, if  $X$  is zero, it means that the cell is not occupied by any breeder).

It is guaranteed that it will be possible for you to go from your starting position to the exit cell through a sequence of moves.

### Output

A single line denoted the minimum possible number of mikemon battles that you have to participate in if you pick a strategy that minimize this number.


Examples

input
5 7 000E0T3 T0TT0T0 010T0T0 2T0T0T0 0T0S000
output
3

input
1 4 SE23
output
2

Note

The following picture illustrates the first example. The blue line denotes a possible sequence of moves that you should post in your blog:

The three breeders on the left side of the map will be able to battle you — the lone breeder can simply stay in his place until you come while the other two breeders can move to where the lone breeder is and stay there until you come. The three breeders on the right does not have a way to battle you, so they will stay in their place.

For the second example, you should post this sequence in your Blog:

Here's what happens. First, you move one cell to the right.



Then, the two breeders directly to the right of the exit will simultaneously move to the left. The other three breeder cannot battle you so they will do nothing.



You end up in the same cell with 2 breeders, so 2 mikemon battles are conducted. After those battles, all of your opponents leave the forest.



Finally, you make another move by leaving the forest.



## C. Graph Reconstruction

time limit per test: 3 seconds  
memory limit per test: 256 megabytes  
input: standard input  
output: standard output

I have an undirected graph consisting of  $n$  nodes, numbered 1 through  $n$ . Each node has at most two incident edges. For each pair of nodes, there is at most an edge connecting them. No edge connects a node to itself.

I would like to create a new graph in such a way that:

- The new graph consists of the same number of nodes and edges as the old graph.
- The properties in the first paragraph still hold.
- For each two nodes  $u$  and  $v$ , if there is an edge connecting them in the old graph, there is no edge connecting them in the new graph.

Help me construct the new graph, or tell me if it is impossible.

### Input

The first line consists of two space-separated integers:  $n$  and  $m$  ( $1 \leq m \leq n \leq 10^5$ ), denoting the number of nodes and edges, respectively. Then  $m$  lines follow. Each of the  $m$  lines consists of two space-separated integers  $u$  and  $v$  ( $1 \leq u, v \leq n; u \neq v$ ), denoting an edge between nodes  $u$  and  $v$ .

### Output

If it is not possible to construct a new graph with the mentioned properties, output a single line consisting of -1. Otherwise, output exactly  $m$  lines. Each line should contain a description of edge in the same way as used in the input format.

### Examples

<b>input</b>
8 7 1 2 2 3 4 5 5 6 6 8 8 7 7 4
<b>output</b>
1 4 4 6 1 6 2 7 7 5 8 5 2 8

<b>input</b>
3 2 1 2 2 3
<b>output</b>
-1

<b>input</b>
5 4 1 2 2 3 3 4 4 1
<b>output</b>
1 3 3 5 5 2 2 4

### Note

The old graph of the first example:



A possible new graph for the first example:



In the second example, we cannot create any new graph.

The old graph of the third example:



A possible new graph for the third example:



## D. The Evil Temple and the Moving Rocks

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

*Important: All possible tests are in the pretest, so you shouldn't hack on this problem. So, if you passed pretests, you will also pass the system test.*

You are an adventurer currently journeying inside an evil temple. After defeating a couple of weak monsters, you arrived at a square room consisting of tiles forming an  $n \times n$  grid, surrounded entirely by walls. At the end of the room lies a door locked with evil magical forces. The following inscriptions are written on the door:

*The sound of clashing rocks will awaken the door!*

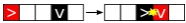
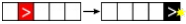
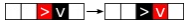
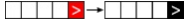
Being a very senior adventurer, you immediately realize what this means. In the room next door lies an infinite number of magical rocks. There are four types of rocks:

- '^': this rock moves upwards;
- '<': this rock moves leftwards;
- '>': this rock moves rightwards;
- 'v': this rock moves downwards.

To open the door, you first need to place the rocks on some of the tiles (one tile can be occupied by at most one rock). Then, you select a single rock that you have placed and activate it. The activated rock will then move in its direction until it hits another rock or hits the walls of the room (the rock will not move if something already blocks it in its chosen direction). The rock then deactivates. If it hits the walls, or if there have been already  $10^7$  events of rock becoming activated, the movements end. Otherwise, the rock that was hit becomes activated and this procedure is repeated.

If a rock moves at least one cell before hitting either the wall or another rock, the hit produces a sound. The door will open once the number of produced sounds is at least  $X$ . It is okay for the rocks to continue moving after producing  $X$  sounds.

The following picture illustrates the four possible scenarios of moving rocks.

- Moves at least one cell, then hits another rock. A sound is produced, the hit rock becomes activated.  

- Moves at least one cell, then hits the wall (i.e., the side of the room). A sound is produced, the movements end.  

- Does not move because a rock is already standing in the path. The blocking rock becomes activated, but no sounds are produced.  

- Does not move because the wall is in the way. No sounds are produced and the movements end.  


Assume there's an infinite number of rocks of each type in the neighboring room. You know what to do: place the rocks and open the door!

### Input

The first line will consist of two integers  $n$  and  $X$ , denoting the size of the room and the number of sounds required to open the door. There will be exactly three test cases for this problem:

- $n = 5, x = 5$ ;
- $n = 3, x = 2$ ;
- $n = 100, x = 10^5$ .

All of these testcases are in pretest.

### Output

Output  $n$  lines. Each line consists of  $n$  characters — the  $j$ -th character of the  $i$ -th line represents the content of the tile at the  $i$ -th row and the  $j$ -th column, and should be one of these:

- '^', '<', '>', or 'v': a rock as described in the problem statement.
- '.': an empty tile.

Then, output two integers  $r$  and  $c$  ( $1 \leq r, c \leq n$ ) on the next line — this means that the rock you activate first is located at the  $r$ -th row from above and  $c$ -th column from the left. There must be a rock in this cell.

If there are multiple solutions, you may output any of them.

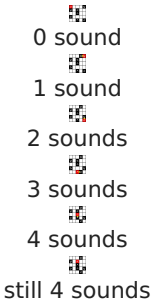
### Examples

input
5 5

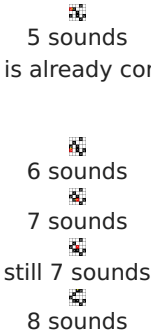
output
<div> <div>&gt;...v</div> <div>v.&lt;..</div> <div>..^..</div> <div>&gt;....</div> <div>..^.&lt;</div> <div>1 1</div> </div>

input
3 2
output
<div> <div>&gt;vv</div> <div>^&lt;.</div> <div>^.&lt;</div> <div>1 3</div> </div>

**Note**  
 Here's a simulation of the first example, accompanied with the number of sounds produced so far.



In the picture above, the activated rock switches between the '^' rock and the '<' rock. However, no sound is produced since the '^' rock didn't move even a single tile. So, still 4 sound.

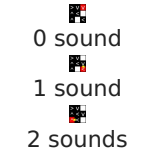


At this point, 5 sound are already produced, so this solution is already correct. However, for the sake of example, we will continue simulating what happens.



And the movement stops. In total, it produces 8 sounds. Notice that the last move produced sound.

Here's a simulation of the second example:



Now, the activated stone will switch continuously from one to another without producing a sound until it reaches the  $10^7$  limit, after which the movement will cease.

In total, it produced exactly 2 sounds, so the solution is correct.



## E. Evil

time limit per test: 3 seconds  
memory limit per test: 256 megabytes  
input: standard input  
output: standard output

There are  $n$  cities on a two dimensional Cartesian plane. The distance between two cities is equal to the Manhattan distance between them (see the Notes for definition). A Hamiltonian cycle of the cities is defined as a permutation of all  $n$  cities. The length of this Hamiltonian cycle is defined as the sum of the distances between adjacent cities in the permutation plus the distance between the first and final city in the permutation. Please compute the longest possible length of a Hamiltonian cycle of the given cities.

### Input

The first line contains an integer  $n$  ( $3 \leq n \leq 10^5$ ). Then  $n$  lines follow, each consisting of two integers  $x_i$  and  $y_i$  ( $0 \leq x_i, y_i \leq 10^9$ ), denoting the coordinates of a city. All given points will be distinct.

### Output

A single line denoting the longest possible length of a Hamiltonian cycle of the given cities. You should not output the cycle, only its length.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

### Examples

input
4 1 1 1 2 2 1 2 2
output
6

### Note

In the example, one of the possible Hamiltonian cycles with length 6 is (1, 1) (1, 2) (2, 1) (2, 2). There does not exist any other Hamiltonian cycle with a length greater than 6.

The Manhattan distance between two cities  $(x_i, y_i)$  and  $(x_j, y_j)$  is  $|x_i - x_j| + |y_i - y_j|$ .