Problem C. lemming

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

small 5 points *2 minute timeout*	The contest is finished.
large 14 points *10 minute timeout*	The contest is finished.

Problem

Lemming

Our Distributed Code Jam team has a pet lemming, Larry, who likes to follow instructions. To give Larry some exercise, we place him on a table with grid of cells, each of which contains an arrow pointing either up, down, left, or right. Larry will always move one unit in the direction indicated by his current cell. If this takes him off the edge of the table, he falls harmlessly onto some padding. Otherwise, he follows the direction indicated by his new cell, and so on, possibly continuing in an infinite loop.

We don't want Larry to fall off the table or exercise forever, so we want to turn one or more of the existing cells on the table into blank *pickup points*. If Larry starts on or reaches a pickup point, we pick him up and the exercise period is over.

What is the minimum number of cells that we need to change into pickup points to ensure that no matter where Larry is initially placed on the grid, he will eventually be picked up, instead of falling off the table or exercising forever?

Input

The input library is called "lemming"; see the sample inputs below for examples in your language. It defines three methods:

- **GetRows()**:
  - Takes no argument.
  - Returns a 64-bit integer: the number of rows in the input grid.
  - Expect each call to take 0.06 microseconds.
- **GetColumns()**:
  - Takes no argument.
  - Returns a 64-bit integer: the number of columns in the input grid.
  - Expect each call to take 0.06 microseconds.
- **GetDirection(r, c)**:
  - Takes two 64-bit integers in the ranges $0 \le r <$ GetRows(), $0 \le c <$ GetColumns().
  - Returns a character: the contents of the cell at row r and column c of the input grid. The character is one of ^ (ASCII code 94), lowercase v, <, > which represents up, down, left and right respectively. Rows are numbered from top to bottom. Columns are numbered from left to right, as explained above.
  - Expect each call to take 0.06 microseconds.

Output

Output one line with a single integer: the minimum number of pickup points that you need to create.

Limits

Number of nodes: 100 **(for both the Small and Large datasets)**.
Time limit: 15 seconds.
Memory limit per node: 1 GB.
Maximum number of messages a single node can send: 1000.
Maximum total size of messages a single node can send: 8 MB.
$1 \le$ GetRows() $\le 30{,}000$.
$1 \le$ GetColumns() $\le 30{,}000$.

Small dataset

GetDirection(r, c) is one of the characters v, <, >, for all r and c.

Large dataset

GetDirection(r, c) is one of the characters ^, v, <, >, for all r and c.

Sample

Input	Output
See input files below.	For sample input 1: 6 For sample input 2: 8 For sample input 3: 4

For ease of reading, these are the input matrices in the samples:

```
<v><
<<v>
>><>

><><><><><><><>

<v<
>>^
v>>
^^^
```

Note that the last sample case would not appear in the Small dataset.

Sample input libraries:
Sample input for test 1: lemming.h [CPP] lemming.java [Java]
Sample input for test 2: lemming.h [CPP] lemming.java [Java]
Sample input for test 3: lemming.h [CPP] lemming.java [Java]