TASK	Board	Adriatic	Watering
type	batch	batch	output only
time limit (per test run)	0.2 seconds	2 seconds	-
memory limit (per test run)	256 MB	256 MB	-
points	100	100	100
	300		

BOARD

Mirko and Slavko have a new board game. The game board resembles a complete infinite binary tree. More precisely, the board consists of nodes and two-way roads connecting them. The root node is located at the top of the board and we say it is at *level zero*. Each node has exactly two children, the *left child* and the *right child*, located in the lower-left and the lower-right directions of the parent node. The level of a child node is one greater than the level of the parent node. In addition to roads connecting a parent node with its children, there are roads connecting all of the nodes at a particular level – for each level, starting from the leftmost node, there is a road connecting each node to the next node to the right on the same level.

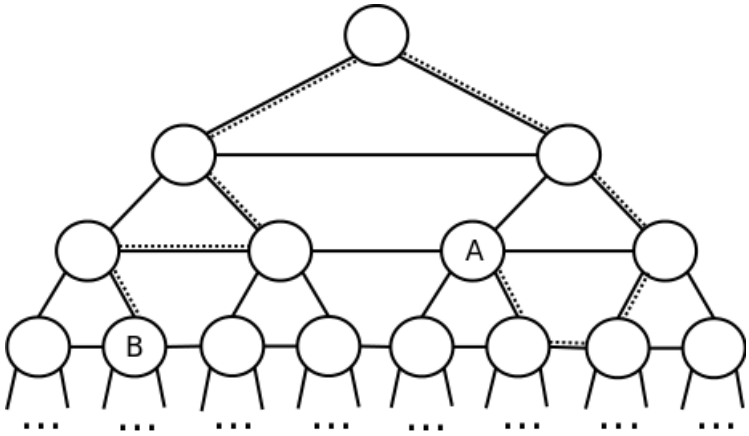


Figure 1: The second test example below

Each *path* through the game board is a sequence of steps, each moving from a node to a different node via a single road. Each step can be described by a single character as follows:

- character '1' describes moving from a node to its left child,
- character '2' describes moving from a node to its right child,
- character 'U' describes moving from a node to its parent,
- character 'L' describes moving from a node to the next node to the left on the same level,
- character 'R' describes moving from a node to the next node to the right on the same level.

For example, if we were to start at the root node and take the sequence of steps '221LU' we would end up in the node denoted with the letter 'A' in the figure above.

TASK

Write a program that will, given two nodes on the board, find the smallest number of steps needed to go from one node to the other. The two nodes are given by specifying paths from the root node to them. If the two paths lead to the same node, the answer is zero.

INPUT

The first line of input contains a sequence of at most 100 000 characters – the path from the root to the first node.

The second line of input contains a sequence of at most 100 000 characters – the path from the root to the second node.

The two paths will be valid (it will be possible to make every move in both sequences).

OUTPUT

The first and only line of output should contain a single integer - the smallest number of steps needed to go from one node to the other.

GRADING

Let *D* be the smallest integer such that both input paths only visit nodes whose levels are at most *D*.

- In test cases worth a total of 20 points, *D* is at most 10.
- In test cases worth a total of 40 points, *D* is at most 50.
- In test cases worth a total of 70 points, *D* is at most 1000.

DETAILED FEEDBACK WHEN SUBMITTING

During the contest, you may select up to 50 submissions for this task to be evaluated on a part of the official test data. When the results are ready, a summary of the results will be available on the contest system.

EXAMPLES

input	**input**	**input**
111RRRRRRR 222	221LU 12L2	11111 222222
output	**output**	**output**
0	3	10

ADRIATIC

"The land of a thousand islands" was an official motto of Croatian tourism in the mid nineteen-nineties. While the motto is technically inaccurate (there are slightly more than 1000 islands in Croatia), it is true that island hopping (sailing from island to island) is a popular summer activity.

For the purpose of this task, the map of the Adriatic sea is a grid consisting of unit squares organized into 2500 rows and 2500 columns. Rows are numbered 1 through 2500, north to south, while columns are numbered 1 through 2500, west to east. There are N islands in the sea, numbered 1 through N and each island is located inside some unit square of the grid. The location of island K is given by the coordinates of the corresponding grid square – its row number R_K and its column number C_K. Finally, no two islands have the same location.

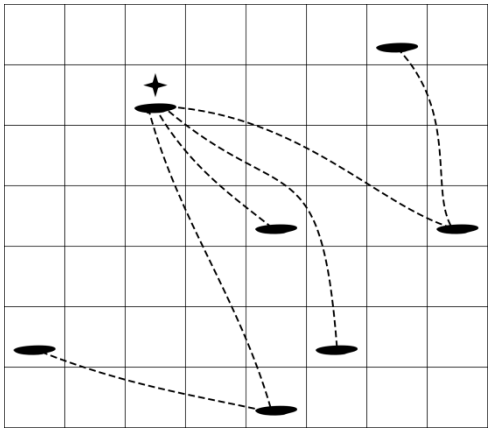


Figure 1: Map corresponding to the first test example below

Due to winds and sea currents, it is possible to sail directly from an island only to those islands that are located in the general northwest or southeast directions. More precisely, it is possible to sail from island A to island B **in one hop** if either both $R_A < R_B$ and $C_A < C_B$ hold or both $R_A > R_B$ and $C_A > C_B$ hold. Note that the distance between the two islands or the presence of other islands between them does not affect the possibility of hopping from one to the other. If it is not possible to hop directly from A to B, it might be possible to sail from A to B via other islands using some sequence of hops. The *sailing distance* from A to B is defined as the smallest number of hops required to sail from A to B.

For example, in the figure above, starting from the island at row 2, column 3, we can hop to four other islands while the sailing distance to the remaining two islands is two.

TASK

A sailing congress is being planned and the organizers are considering each of the islands as a possible location for the congress. When considering a candidate island they would like to know: if every other island sends a single sailboat, what is the **smallest total number of hops** required in order for **all sailboats to reach the candidate island**, or equivalently, what is the sum of sailing distances from all other islands to the candidate island. Write a program that will, given the locations of N islands, for each island K, calculate the sum of sailing distances from all other islands to island K.

Test data will be such that, for all islands A and B it is possible to sail from A to B using some sequence of hops.

INPUT

The first line of input contains an integer N ($3 \leq N \leq 250\,000$), the number of islands. The following N lines contain the locations of the islands. Each location is a pair of integers between 1 and 2500 (inclusive), the row and column numbers, respectively.

OUTPUT

The output should contain N lines. For each island, in the same order they were given in the input, output the sum of sailing distances from all other islands on a single line.

GRADING

- In test cases worth a total of 25 points, N will be at most 100.
- In test cases worth a total of 50 points, N will be at most 1500.
- In test cases worth a total of 60 points, N will be at most 5000.
- In test cases worth a total of 80 points, N will be at most 25000.

EXAMPLES

input

```
7
1 7
7 5
4 5
4 8
6 6
6 1
2 3
```

output

```
16
11
12
11
12
16
8
```

input

```
4
1 1
2 3
3 2
4 4
```

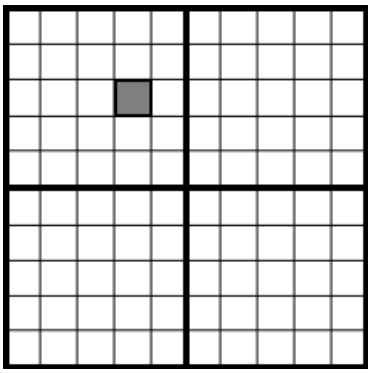
output

```
3
4
4
3
```

WATERING

Sara is a dedicated farmer owning a large rectangular piece of land. Her land has a number of *cells* nicely arranged into a grid consisting of 5**R* rows and 5**C* columns. Furthermore, there is a horizontal fence across the width of the board after every fifth row and a vertical fence across the height of the board after every fifth column. The fences divide the land into *R***C* 5x5 areas called *fields*.

The two most common problems Sara faces are birds and droughts. To combat annoying crop-eating birds, some fields are equipped with a scarecrow. A scarecrow (if present) occupies a **single cell** and there can be **at most one scarecrow in each 5x5 field**.

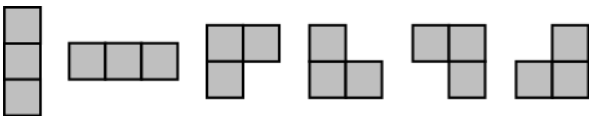


```
.....|.....
.....|.....
...#.|.....
.....|.....
.....|.....
-----+-----
.....|.....
.....|.....
.....|.....
.....|.....
.....|.....
```

```
aaacc|dxxxa
bbbce|dyyya
ddd#e|dzzza
ccbae|fccbb
cbbaa|ffcdb
-----+---_-
ssrrr|tttdd
saaax_xxeee
yxbbb|zdaaa
yxccc|zdbbb
yxddd|zdccc
```

Figure 1: Example layout of Sara's land, its textual representation and a valid sprinkler arrangement

During droughts, which can last for months, Sara uses sprinklers to keep her crops watered. Each of her sprinklers has three nozzles: one main nozzle and two side nozzles. It occupies **exactly three cells** and waters all of them. The side nozzles always occupy exactly **two cells that are adjacent** (up, down, left or right) to the main nozzle. Hence, a single sprinkler is always in one of the following configurations:



Sara wants to put sprinklers on her land in such a way that there is **exactly one sprinkler** on every cell that is not occupied with a scarecrow. A cell containing a scarecrow **must not** contain a sprinkler nozzle. In addition, nozzles must not be placed outside Sara's land.

The three cells watered by a single sprinkler need not belong to the same 5x5 field: they may also belong to the neighboring fields. In that case, Sara has to **drill a hole in the fence**, **between the two cells in neighboring fields watered by the same sprinkler**. Drilling a hole is difficult for Sara: she does not want to drill many.

TASK

Given the description of Sara's land you need to produce a valid configuration of sprinklers to water it. If you succeed, your score will depend on the total number of holes that need to be made in the fences – see the Grading section for details.

This is an output only task. You will be given 10 input files and you only need to produce the matching output files. You may download the input files from the contest system, on the page labeled 'Tasks'.

You need to submit each output file separately using the contest system. When submitting, the contest system will check the format of your output file. If the format is valid, the output file will be **graded and the score reported**; otherwise, the contest system will report an error. Hence, you will get **full feedback** for the output files submitted for this task.

Test data will be such that a solution always exists. If there is more than one solution, you may submit any one.

INPUT

The first line of the input contains a pair of integers R and C ($1 \le R, C \le 100$) – the size of Sara's land as described above.

$6*R-1$ lines follow, each containing a sequence $6*C-1$ characters. They represent Sara's fields and fences between them. The fence itself is also represented with characters even though the fence is actually infinitely thin.

A single cell is represented by a single character. The dot character '.' represents an empty cell, while the '#' character (ASCII 35) represents a scarecrow. Vertical fences are represented by the '|' character (ASCII 124) and the horizontal fences with the '−' character (minus). The '+' character denotes an intersection of fences.

OUTPUT

The output file should contain the textual representation of the field with a valid sprinkler arrangement in the same format as the input file. Each hole in the fence should be denoted by the underscore character '_'. All empty cells (dots) from the input file should be replaced by lowercase letters 'a' – 'z' so that the following rules are satisfied:

1. Any three cells watered by the same sprinkler are denoted by the same letter, even if not all of them are in the same 5x5 field.
2. If two adjacent cells **in the same field** are watered by different sprinklers, they must be denoted by different letters.
3. If two adjacent cells **in different fields** are watered by **different sprinklers** and there is a **hole in the fence** between them, they must be denoted by different letters.
4. It is allowed for adjacent cells that belong to different fields to be denoted by the same letter, as long as all the previous rules are satisfied.

GRADING

Each test case is worth 10 points. If the watering configuration is not valid, you will get zero points for that test case. If the configuration is valid, the solution will be graded as follows:

- If the number of holes in fences is not greater than $R*C$, your score is 10 points.
- Otherwise, your score is 5 points.
- In 4 out of 10 official test inputs, there will be a scarecrow in every field.

EXAMPLE

input

```
2 2
.....|.....
.....|.....
...#.|.....
.....|.....
.....|.....
-----+-----
.....|.....
.....|.....
.....|.....
.....|.....
.....|.....
```

output

```
aaacc|dxxxa
bbbce|dyyya
ddd#e|dzzza
ccbae|fccbb
cbbaa|ffcdb
-----+---_-
ssrrr|tttdd
saaax_xxeee
yxbbb|zdaaa
yxccc|zdbbb
yxddd|zdccc
```