Write a code that specifies how many button presses are needed to write the given text with a TV remote control on a virtual keyboard. Keyboard layout looks like this:

**a b c d e 1 2 3**

**f g h i j 4 5 6**

**k l m n o 7 8 9**

**p q r s t . @ 0**

**u v w x y z \_ /**

The cursor starts on the letter **a**, we can navigate on the keyboard with **up, down, left and right** arrows. We must press **OK** each time to select character, the cursor does not wrap on edges and does not return to start after press.

**Input**

* single **string** composed with characters available on the keyboard
* 0 <**string**< 32 chars

**Output**

* the **number** of button presses needed to write the string

**Example unit tests**

assertEquals(6, App.solve("agh")); // OK - b - g - OK - h - OK

At the foot of a mountain slope, there is a small village. One day a team of scientists arrived and they told the villagers that they are in danger of rock avalanches and thus should evacuate. The villagers, however, are very attached to their homes and refuse to leave. Given the situation, the researchers decide that it is necessary to create a simulation of the disaster so as to demonstrate to the villagers which houses will be crushed by rocks. This task has been assigned to you.

Write a function that analyses a given image of the slope and indicates how many rocks will fall on each of the village houses below. The slope is represented by a two-dimensional array of characters. The available characters are:

* the letter **o** - indicating a rock
* a dot (**.**) - a place that a rock can fall freely onto
* an underscore (**\_**) - a place from which a rock cannot move - so the rock is stationary
* a slash (**/**) - a place from which a rock moves one unit left
* a backslash (**\**) - a place from which a rock moves one unit right

For simplicity, rocks can only occur at the highest level. Also no **\**or **/**symbol will be put on far most right or left side of the slope so no rock will be thrown off.

**Input**

* **Array** (or **Vector** in c++) of **strings** each indicating a single slope level
* 0 <**array**< 8 strings**of the same size**
* 0 <**string**< 64 chars

**Output**

* **Array** (or **Vector**) of **integers** indicating the number of rocks that have fallen to a given house.

**Example**

For a slope given with:

**o o o**

**\_ \ .**

the function should return an array like this:

**0 0 2**

The leftmost rock was stationary, the next one moved right and fell, along with the third rock, onto the rightmost house.

**Example unit tests**

assertArrayEquals(new int[]{0,0,0}, App.solve(new char[][]{{'.','o','o'},{'\_','\_','\_'}}));

Write a code that will tell Kevin whether he is home alone or not! The code should be able to receive a 2D character matrix illustrating Kevin's house. The matrix consists of the following characters:

* **#** - standing for the walls of a house
* **K** - standing for Kevin
* **o** - standing for a thief
* (space) - standing for a free space

All the walls in the matrix must form some sort of closed building(s) with square corners. Sometimes there can be more than one building in the matrix, but only one of them is Kevin's house - the one he currently lives in.

**Input**

* **Array** (or **Vector** in c++) of **strings** forming Kevin's house
* 0 <**array**< 32 strings**of the same size**
* 0 <**string**< 128 chars

**Output**

* Single**boolean**value

**Example**

**o o #######**

**########## # #**

**###### # o # #**

**# K # ########### #**

**####### # # o #**

**######### #################**

As we can see here, Kevin is home alone in this example. The method should return **true**.

**Example unit tests**

char[][] house = {

" o o #######".toCharArray(),

" ########## # #".toCharArray(),

"###### # o # #".toCharArray(),

"# K ############### #".toCharArray(),

"####### #".toCharArray(),

" #############################".toCharArray()

};

assertEquals(true, App.isAlone(house));