Bididibus is an ancient two-dimensional universe populated by the Bididibusians. Bididibusians have their own 2D water, 2D mountains, 1D television, and even an amazing new technology of 2D cinemas!

These days, some Bididibusian scholars are warning the public about the risks of *global raining*. Such climate catastrophe would fill with water all the valleys of the universe.

Bididibus consists of blocks with uniform width and height. Bididibusian orography is described by a series of symbols, from left to right, indicating that the land rises one unit (represented by "/"), descends one unit (represented by "\"), or it is flat (represented by "_").

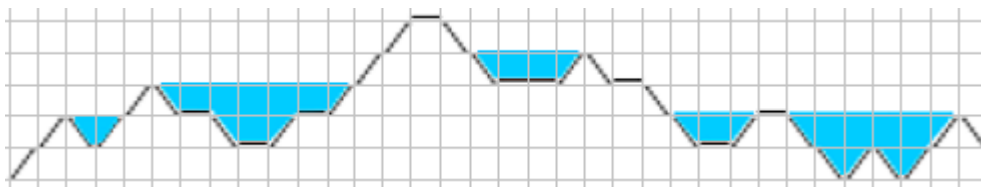
For example, we can have a 2D universe like this:

```
             _
            / \
           /   \__/\_
    /\_   _/         \   _
 /\/   \_/            \_/ \    /\
/                          \/\/
```

It goes without saying that native geographers are big fans of ASCII art. They also use a simplified representation, where each column (from left to right) is replaced by the symbol it contains. For example, the simplified representation of the previous universe would be:

```
//\//\_\_/_///_\\__/\_\\_/_\\/\//\
```

After the global rain, water would fill all the blocks of the valleys. If we represent water areas in blue, we would have the following:



Observe that a block with '/' or '\' partially filled with water corresponds to 1/2 units of water. A block completely filled with water corresponds to 1 unit of water. For example, in the previous universe we have a total of 21 units of water.

Given a simplified representation of a universe, your task is to compute the units of water that we would have after the global rain.

Input

The first line of the input contains an integer indicating the number of test cases.

For each test case, there is a line which can contain three possible symbols: '/', '\', '_'. There will be at most 10000 symbols in each line.

Output

For each test case, the output should contain a single integer indicating the units of water of the corresponding case.

Sample Input

```
4
//\//\_\_/_///_\\__/\_\\_/_\\/\//\
////\\\\\\\\\
\/
\\\\\\\\\\/////////
```

Sample Output

```
21
0
1
100
```