Teaser Problems

September 5, 2013

Instructions

These problems are taken from the 2006 Mid Central North American Regionals, We encourage you to give them a try. When you think that you have a solution, you can use the ACM-ICPC Live Archive at `https://icpcarchive.ecs.baylor.edu/index.php` to test your solution. Note that you must be logged in to submit, so if you don't have an account go ahead and register.

To find the problems In this set, you can use this link `https://icpcarchive.ecs.baylor.edu/index.php?option=com_onlinejudge&Itemid=8&category=264` They are the first 3 problems listed on that page.

Notes on submitting

- The online judge accepts solutions in C,C++ and Java (and Pascal, but I doubt many of you will use that).
- All Questions take input from standard in (cin in C++, System.in for Java) and take output from standard out (cout in C++, System.out for Java)
- For Java, The class should be called Main and should be public

3594 - Quicksum

North America - Mid Central - 2006/2007

A checksum is an algorithm that scans a packet of data and returns a single number. The idea is that if the packet is changed, the checksum will also change, so checksums are often used for detecting transmission errors, validating document contents, and in many other situations where it is necessary to detect undesirable changes in data.

For this problem, you will implement a checksum algorithm called Quicksum. A Quicksum packet allows only uppercase letters and spaces. It always begins and ends with an uppercase letter. Otherwise, spaces and letters can occur in any combination, including consecutive spaces.

A Quicksum is the sum of the products of each character's position in the packet times the character's value. A space has a value of zero, while letters have a value equal to their position in the alphabet. So, `A`=1, `B`=2, etc., through `Z`=26. Here are example Quicksum calculations for the packets ``ACM'' and ``MID CENTRAL'':

`ACM`: 1 * 1 + 2 * 3 + 3 * 13 = 46

`MID CENTRAL`: 1 * 13 + 2 * 9 + 3 * 4 + 4 * 0 + 5 * 3 + 6 * 5 + 7 * 14 + 8 * 20 + 9 * 18 + 10 * 1 + 11 * 12 = 650

Input

The input consists of one or more packets followed by a line containing only `#` that signals the end of the input. Each packet is on a line by itself, does not begin or end with a space, and contains from 1 to 255 characters.

Output

For each packet, output its Quicksum on a separate line in the output.

Sample Input

```
ACM
MID CENTRAL
REGIONAL PROGRAMMING CONTEST
ACN
A C M
ABC
BBC
#
```

Sample Output

```
46
650
4690
49
75
```

3595 - Linear Pachinko

North America - Mid Central - 2006/2007

This problem is inspired by Pachinko, a popular game in Japan. A traditional Pachinko machine is a cross between a vertical pinball machine and a slot machine. The player launches small steel balls to the top of the machine using a plunger as in pinball. A ball drops through a maze of pins that deflect the ball, and eventually the ball either exits at a hole in the bottom and is lost, or lands in one of many gates scattered throughout the machine which reward the player with more balls in varying amounts. Players who collect enough balls can trade them in for prizes.

For the purposes of this problem, a linear Pachinko machine is a sequence of one or more of the following: holes (``."), floor tiles (``_"), walls (``|"), and mountains (``/\"). A wall or mountain will never be adjacent to another wall or mountain. To play the game, a ball is dropped at random over some character within a machine. A ball dropped into a hole falls through. A ball dropped onto a floor tile stops immediately. A ball dropped onto the left side of a mountain rolls to the left across any number of consecutive floor tiles until it falls into a hole, falls off the left end of the machine, or stops by hitting a wall or mountain. A ball dropped onto the right side of a mountain behaves similarly. A ball dropped onto a wall behaves as if it were dropped onto the left or right side of a mountain, with a 50% chance for each. If a ball is dropped at random over the machine, with all starting positions being equally likely, what is the probability that the ball will fall either through a hole or off an end? For example, consider the following machine, where the numbers just indicate character positions and are not part of the machine itself:

```
123456789
/\.|__/\.
```

The probabilities that a ball will fall through a hole or off the end of the machine are as follows, by position: 1 = 100%, 2 = 100%, 3 = 100%, 4 = 50%, 5 = 0%, 6 = 0%, 7 = 0%, 8 = 100%, 9 = 100%. The combined probability for the whole machine is just the average, which is approximately 61.111%.

Input

The input consists of one or more linear Pachinko machines, each 1â 79 characters long and on a line by itself, followed by a line containing only "#" that signals the end of the input.

Output

For each machine, compute as accurately as possible the probability that a ball will fall through a hole or off the end when dropped at random, then output a single line containing that percentage truncated to an integer by dropping any fractional part.

Sample Input

```
/\.|__/\.
_._/\_|.__/\./\_
...
___
./\.
```

3596 - Surprising Strings

North America - Mid Central - 2006/2007

The D-pairs of a string of letters are the ordered pairs of letters that are distance D from each other. A string is D-unique if all of its D-pairs are different. A string is surprising if it is D-unique for every possible distance D.

Consider the string `ZGBG`. Its 0-pairs are `ZG`, `GB`, and `BG`. Since these three pairs are all different, `ZGBG` is 0-unique. Similarly, the 1-pairs of `ZGBG` are `ZB` and `GG`, and since these two pairs are different, `ZGBG` is 1-unique. Finally, the only 2-pair of `ZGBG` is `ZG`, so `ZGBG` is 2-unique. Thus `ZGBG` is surprising. (Note that the fact that `ZG` is both a 0-pair and a 2-pair of `ZGBG` is irrelevant, because 0 and 2 are different distances.)

Input

The input consists of one or more nonempty strings of at most 79 uppercase letters, each string on a line by itself, followed by a line containing only an asterisk that signals the end of the input.

Output

For each string of letters, output whether or not it is surprising using the exact output format shown below.

Acknowledgement: This problem is inspired by the ``Puzzling Adventures" column in the December 2003 issue of Scientific American.

Sample Input

```
ZGBG
X
EE
AAB
AABA
AABB
BCBABCC
*
```

Sample Output

```
ZGBG is surprising.
X is surprising.
EE is surprising.
AAB is surprising.
AABA is surprising.
AABB is NOT surprising.
BCBABCC is NOT surprising.
```

Mid Central 2006-2007