Code Jam to I/O for Women 2020 - Code Jam to I/O for Women

Interleaved Output: Part 1

Problem

*You do not need to read the **Interleaved Output: Part 2** problem to be able to solve this problem. Both Part 1 and Part 2 have the same first two paragraphs (not including this informational text). We have underlined the critical difference between the two parts.*

On a distant moon of Jupiter, some developer conference events are about to happen! They are called `IO` (uppercase I, uppercase O), `Io` (uppercase I, lowercase o), `iO` (lowercase I, uppercase O), and `io` (lowercase I, lowercase O).

The best way to advertise an event is by using special computers that print the event's name one character at a time, with the output appearing on a digital display. Each such computer only knows the name of one event, and is programmed to print its event's name zero or more times. For example, a computer programmed to print `IO` twice prints an `I`, followed by an `O`, followed by an `I`, followed by an `O`, for a final string of `IOIO`.

You know that the conference organizers are using these computers, but you do not know how many computers are advertising each event. For each event, there may be any number of computers (including zero) programmed to print the event's name. Moreover, the computers are not necessarily all programmed to print the same number of times. For example, it is possible that there are three computers programmed to print `Io` once each, and one computer programmed to print `Io` twice.

The computers have all finished printing, but unfortunately, they all printed to the same display! Because the computers printed concurrently, event names in the final output string may be interleaved. You are considering the possible ways in which that string could have been produced.

For example, the string `IiOioIoO` could have been produced by two computers, as follows:

- A: programmed to print `Io` twice
- B: programmed to print `iO` twice

```
index:  1 2 3 4 5 6 7 8
A:      I . . . o I o .
B:      . i O i . . . O
string: I i O i o I o O
```

In this interpretation, the `Io` event was advertised twice, the `iO` event was advertised twice, and the other two events were not advertised at all.

But the string could have also been produced by three computers:

- A: programmed to print `IO` twice
- B: programmed to print `io` once
- C: programmed to print `io` once

```
index:  1 2 3 4 5 6 7 8
A:      I . O . . I . O
B:      . i . . o . . .
```

```
C:      . . . i . . o .
string: I i O i o I o O
```

In this interpretation, the `IO` event was advertised twice, the `io` event was advertised twice, and the other two events were not advertised at all. Notice that this interpretation required two computers printing `io`; there could not have been just one computer printing `io` twice, because it would have had to print `i` twice in a row, which is not allowed.

Given a final output string, what is the maximum possible number of times that the event `IO` could have been advertised?

It is guaranteed that the string has at least one valid interpretation. For example, `oI` and `IOI` are not valid inputs.

Input

The first line of the input gives the number of test cases, **T**. **T** lines follow; each represents a single test case. Each case consists of a string **S** containing only the characters from the set `I`, `O`, `i`, and `o`.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the maximum number of times `IO` could have been advertised, as described above.

Limits

Time limit: 20 seconds per test set.
Memory limit: 1GB.
1 ≤ **T** ≤ 100.
The length of **S** is even.
For each prefix S' of **S**, the number of `i` characters plus the number of `I` characters in S' is not less than the number of `o` characters plus the number of `O` characters in S'.
The number of `i`s plus the number of `I`s in **S** is equal to the number of `o`s plus the number of `O`s in **S**.
(Notice that the above three conditions guarantee that there is at least one interpretation of **S** that is consistent with the rules in the problem statement.)

Test set 1 (Visible Verdict)

2 ≤ the length of **S** ≤ 8.

Test set 2 (Hidden Verdict)

2 ≤ the length of **S** ≤ 100.

Sample

Input	Output

```
5
IiOioIoO
IiOOIo
IoiOiO
io
IIIIOOOO
```

```
Case #1: 2
Case #2: 1
Case #3: 0
Case #4: 0
Case #5: 4
```

Sample Case #1 is the one described in the problem statement. We saw that there is an interpretation in which `IO` was advertised twice. There are only two `I`s and two `O`s in the string, so the answer cannot be larger than this.

In Sample Case #2, notice that it is not possible that `IO` was advertised twice. The only possible interpretations are as follows:

- A: programmed to print `IO` once
- B: programmed to print `iO` once
- C: programmed to print `Io` once

```
index:  1 2 3 4 5 6
A:      I . . O . .
B:      . i O . . .
C:      . . . . I o
string: I i O O I o
```

or the same but with

```
index:  1 2 3 4 5 6
A:      I . O . . .
B:      . i . O . .
C:      . . . . I o
string: I i O O I o
```

In either of these interpretations, `IO` was advertised only once.

In Sample Case #3, there is no possible interpretation in which `IO` was advertised. There must have been one computer programmed to print `Io` once, and either one computer printing `iO` twice, or two computers printing `iO` once each.

In Sample Case #4, notice that it is possible that `I` and/or `O` might not even show up in the string.

In Sample Case #5, an interpretation in which `IO` was advertised four times requires four computers, each of which was programmed to print `IO` once.