Problem 3—Just Ken

Ken doesn't enjoy being defined as being second best to Barbie. So, Ken's goal is to hide in a word grid and be free of Barbie. KEN is free of BARBIE if KEN is found in the grid horizontally, vertically, or diagonally without any instance of BARBIE touching KEN. BARBIE is touching KEN if any square in the BARBIE touches any square in the KEN, horizontally, vertically, or diagonally, or if they have a square in common (the E).

For example,

```
KENKKN
NNNNEK
BARBIE
```

There is only one BARBIE, on the bottom row. There are two KENs, on the top row and backwards on the second row. The top KEN is free of BARBIE. The middle row KEN is not.

```
KENKKN
BARBIE
NNNNEK
```

There are three KENS in this one and none of them are free of BARBIE.

You are to compute the number of KENs that are free of BARBIE in a given grid.

INPUT SPECIFICATION. Each input case consists of two unsigned decimal integers representing the number of rows and columns (*r* and *c*), respectively, in the grid with one space between them and **<EOLN>** following the *c*. There will follow *r* lines, each consisting entirely of *c* uppercase letters and terminated with **<EOLN>**. "0 0**<EOLN>**" follows the last input case and is not to be processed.

OUTPUT SPECIFICATION. The output cases should be processed in the same order as their respective input cases. Each output case should be of the format "Case *cs*: There are *k* Ken(s) without Barbie." where *cs* is the case number and *k* is the number of KENs in the grid without BARBIE. **<EOLN>** should follow each output case.

SAMPLE INPUT.

```
3·6<EOLN>
KENKKN<EOLN>
BARBIE<EOLN>
NNNNEK<EOLN>
3·6<EOLN>
KENKKN<EOLN>
NNNNEK<EOLN>
BARBIE<EOLN>
0·0<EOLN>
<EOF>
```

SAMPLE OUTPUT.

```
Case·1:··There·are·0·Ken(s)·without·Barbie.<EOLN>
Case·2:··There·are·1·Ken(s)·without·Barbie.<EOLN>
<EOF>
```