Key Task

`keys.c | keys.C | keys.java | keys.p`

The Czech Technical University is rather old — you already know that it celebrates 300 years of its existence in 2007. Some of the university buildings are old as well. And the navigation in old buildings can sometimes be a little bit tricky, because of strange long corridors that fork and join at absolutely unexpected places.

The result is that some first-graders have often difficulties finding the right way to their classes. Therefore, the Student Union has developed a computer game to help the students to practice their orientation skills. The goal of the game is to find the way out of a labyrinth. Your task is to write a verification software that solves this game.

The labyrinth is a 2-dimensional grid of squares, each square is either free or filled with a wall. Some of the free squares may contain doors or keys. There are four different types of keys and doors: blue, yellow, red, and green. Each key can open only doors of the same color.

You can move between adjacent free squares vertically or horizontally, diagonal movement is not allowed. You may not go across walls and you cannot leave the labyrinth area. If a square contains a door, you may go there only if you have stepped on a square with an appropriate key before.

Input Specification

The input consists of several maps. Each map begins with a line containing two integer numbers R and C ($1 \leq R, C \leq 100$) specifying the map size. Then there are R lines each containing C characters. Each character is one of the following:

Character		**Meaning**
Hash mark	`#`	Wall
Dot	`.`	Free square
Asterisk	`*`	Your position
Uppercase letter	`B Y R G`	Blue, yellow, red, or green door
Lowercase letter	`b y r g`	Blue, yellow, red, or green key
Uppercase X	`X`	Exit

Note that it is allowed to have

- more than one exit,
- no exit at all,
- more doors and/or keys of the same color, and
- keys without corresponding doors and vice versa.

You may assume that the marker of your position ("`*`") will appear exactly once in every map.

There is one blank line after each map. The input is terminated by two zeros in place of the map size.

Output Specification

For each map, print one line containing the sentence "`Escape possible in` S `steps.`", where S is the smallest possible number of step to reach any of the exits. If no exit can be reached, output the string "`The poor student is trapped!`" instead.

One step is defined as a movement between two adjacent cells. Grabbing a key or unlocking a door does not count as a step.

Sample Input

```
1 10
*........X

1 3
*#X

3 20
####################
#XY.gBr.*.Rb.G.GG.y#
####################

0 0
```

Output for Sample Input

```
Escape possible in 9 steps.
The poor student is trapped!
Escape possible in 45 steps.
```