B Balloons and Dubloons

Time limit: 2s



Blackbeard, Brownbeard, and Redbeard as team *Segfault go YARRRR*, before Redbeard joined the jury.

Every year, the Be A Pirate Competition is held. This year, Captain Redbeard has become part of the jury. While he does enjoy being a part of the jury, it also means he cannot participate, and thus cannot win the coffer of gold dubloons reserved for the winner.

Now he could, like many of his fellow jury members, pocket a few gold coins every once in a while. But Captain Redbeard has a better plan: his friends (Captain Brownbeard and Captain Blackbeard) are still participating this year, and they agreed to share some of the bounty if he helped them win.

The next task in the competition is finding a treasure on an island. Participants are given a map on which the treasure is marked with a big '`X`'. However, to make things more interesting, there are a number of false '`X`'s to lead pirates astray. Captain Redbeard would like to somehow indicate which treasure marker leads to the actual treasure, but he has no opportunity to directly tell his friends.

But he has a plan! Each jury member will add a single false '`X`' to the map, and he made sure to be last in line. By cleverly placing this final '`X`', he can communicate the location of the real treasure to his friends through a previously established protocol.

If you help Captain Redbeard design and implement such an algorithm, he will award you with one large gold ~~dubloon~~ balloon.

Input

This is a *multi-pass* problem. Your program will be invoked two times.

For each pass, the input consists of:

- One line with a single string, indicating which role your program takes:
  - "`captain`" if you are Captain Redbeard and need to place an additional '`X`'.
  - "`matey`" if you are one of Captain Redbeard's accomplices and need to determine the location of the actual treasure.
- Regardless of the role, one line with two integers h and w ($1 \leq h, w \leq 1000$), the height and width of the treasure map.
- Next follow h lines, each with a string consisting of w characters, describing the treasure map:
  - A character '`~`' indicates water.
  - A character '`.`' indicates unmarked land.
  - A character '`X`' indicates a treasure marker.
  - A character '`#`' indicates a treasure marker pointing to the actual treasure.

If your role is "`captain`", the treasure map contains precisely one '`#`', and it is guaranteed that $n > k^2$ where n is the number of unmarked land tiles, and k is the total number of treasure markers, including the one pointing to the actual treasure.

If your role is "`matey`", the treasure map is the same map that you received as "`captain`", but with the '`#`' replaced by '`X`' and an additional '`X`' placed at the coordinates output by your program as "`captain`".

A testing tool is provided to help you develop your solution.

Output

If your role is "`captain`", output the treasure map from the input, with exactly one piece of unmarked land ('`.`') replaced by an additional treasure marker ('`X`').

If your role is "`matey`", output the treasure map from the input, with exactly one treasure marker ('`X`') replaced by the character '`#`', pointing to the actual treasure.

Sample Case 1

Input	**Pass 1**	**Output**

`captain`
`2 3`
`~~.`
`~.#`
	`~~.`
	`~X#`	

Input	**Pass 2**	**Output**

`matey`
`2 3`
`~~.`
`~XX`
	`~~.`
	`~X#`	

Sample Case 2

Input	**Pass 1**	**Output**

`captain`
`4 5`
`~~.X~`
`.X..~`
`..#..`
`~..~~`
	`~~.X~`
	`.XX.~`
	`..#..`
	`~..~~`	

Input	**Pass 2**	**Output**

`matey`
`4 5`
`~~.X~`
`.XX.~`
`..X..`
`~..~~`
	`~~.X~`
	`.XX.~`
	`..#..`
	`~..~~`	