2006 South Central USA Regional Programming Contest

Enigmatologically Cruciverbalistic

Introduction:

The local ACM Newsletter has decided to start running a crossword puzzle in each issue. Instead of purchasing premade puzzles, however, the editors would like to be able to make puzzles in various shapes and with words of their own choosing. They've turned to you for the development of the puzzles ... and as a programmer, you've decided to turn to your computer and solve the problem once and for all.

A crossword grid looks like this:

```
..#......
#.#......
#########
#.#.#...#
#.#.....#
######..#
#.#......
```

Each continuous row or column of two or more hash marks represents a single word, one letter per mark. Where rows and columns cross, the letter in the shared mark must be the same.

Your goal is to write a program that fills a given grid with a given set of words such that:

- every mark in the grid contains a letter;
- no other cells in the grid contain a letter;
- every word in the word list appears once and only once in the grid with no unused words left over; and
- no words (or other sequences of letters) not in the word list appear in the grid

Words inside of other words (BRIGHT inside of BRIGHTLY, for example) do not count as an appearance of the sub-word. You may assume that every word in the word list is unique in that puzzle. You may also assume that there is at most one possible layout for a given grid and word list.

Input:

The first line of data will be a number representing the number of datasets in the file. For each dataset, the first line consists of two integers w h ($2 <= w, h <= 15$) where w is the width of the puzzle and h is the height. The next h lines of the dataset are a representation of the crossword puzzle grid, as shown above. All word spaces in the puzzle will be at least two characters in length. The next line consists of an integer c ($1 <= c <= 100$) representing the number of words in the crossword. The next c lines contain the words in the

word list.

Output:

For each dataset in the input, output the heading "`Puzzle #x`", where *x* is 1 for the first dataset, 2 for the second, and so on. Then print either "`I cannot generate this puzzle.`" if the puzzle is impossible to generate given that grid and word list, or print a solved representation of the puzzle as shown below.

Sample Input:

```
2
9 7
..#......
#.#......
#########
#.#.#...#
#.#.....#
######..#
#.#......
6
COMPUTE
LAMPSHADE
EDIT
SO
PLAYER
ESTEEM
5 5
#####
#.#.#
#####
#.#.#
#####
6
ADAGE
ANGRY
YEARN
NEEDS
FLUTE
XYZZY
```

Sample Output:

```
Puzzle #1
..C......
P.O......
LAMPSHADE
A.P.O...D
Y.U.....I
ESTEEM..T
R.E......
Puzzle #2
I cannot generate this puzzle.
```