Problem C. Compact Code

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

A maze-like space in underground of the newly opened asteroid was found by a survey a couple of years ago. The project is to investigate the maze more in detail using an exploration robot.

The shape of the maze was fully grasped by a survey with ground penetrating radars. For planning the exploration, the maze is represented as a grid of cells, where the first cell of the first row is the upper left corner of the grid, and the last cell of the last row is the lower right corner.

Each of the grid cell is either vacant, allowing robot's moves to it from an adjacent vacant cell, or filled with rocks, obstructing such moves. The entrance of the maze is located in a cell in the uppermost row and the exit is in a cell in the lowermost row.

The exploration robot is controlled by a program stored in it, which consists of sequentially numbered lines, each containing one of the five kinds of commands described below. The register *pc* specifies the line number of the program to be executed. Each command specifies an action of the robot and a value to be set to *pc*.

The robot starts at the entrance of the maze facing downwards, and the value of *pc* is set to 1. The program commands on the lines specified by *pc* are executed repetitively, one by one, until the robot reaches the exit of the maze.

When the value of *pc* exceeds the number of lines of the program by its increment, it is reset to 1. The robot stops on reaching the exit cell, which is the goal of the project.

As the capacity of the program store for the robot is quite limited, the number of lines of the program should be minimal. Your job is to develop a program with the fewest possible number of lines among those which eventually lead the robot to the exit.

Input

The first line contains two integers n ($2 \le n \le 10$) and m ($2 \le m \le 10$). The maze is represented as a grid with n rows and m columns. The next n lines describe the grid. Each of the lines contains a string of length m corresponding to one grid row. The i-th character of the j-th string, either '`.`', '`#`', '`S`' or '`G`', describes the i-th cell of the j-th row. '`.`' means that the cell is vacant and the robot in one of the four adjacent cells can move to it. '`#`' means that the cell is filled obstructing the robot's moves to it. '`S`' means that the cell is the entrance, and '`G`' means that the cell is the exit. These cells are vacant, of course.

It is known that a program exists that leads the robot to the exit.

The list of commands:

- `GOTO l` — set l to *pc*. The command parameter l is a positive integer less than or equal to the number of lines of the program.
- `IF-OPEN l` — if there is a vacant adjacent cell in its current direction, set l to *pc*; otherwise, that is, facing a filled cell or a border of the grid, increment *pc* by one. The command parameter l is a positive integer less than or equal to the number of lines of the program.
- `FORWARD` — if there is a vacant adjacent cell in its current direction, move there; otherwise, stay in the current cell. In either case, increment *pc* by one.
- `LEFT` — turn 90 degrees to the left without changing the position, and increment *pc* by one.
- `RIGHT` — turn 90 degrees to the right without changing the position, and increment *pc* by one.

Output

The first line of the output should have the number of lines of the program. The commands in the program lines should follow, one per each line, in the order of their line numbers. When the command has a parameter, output only one space between the command name and the parameter.

If more than one appropriate program has the fewest lines, whichever is acceptable.

Examples

standard input	standard output
2 2 .S G#	2 FORWARD LEFT
5 2 S. .. .. .. .G	3 IF-OPEN 3 LEFT FORWARD
2 6 ..S... ..#.G#	4 RIGHT RIGHT FORWARD GOTO 2