**Rescue**

# Program Name: Rescue.java Input File: rescue.dat

You want to use your computer science prowess to help the world. You heard the other day about an earthquake and how people can get trapped and lost in rubble. You decide to create a robot and use it to search rubble for survivors. Since time is of the essence, you need to know how long it takes to get from one place to another.

**Input**

The first line will contain int n, the number of test cases. For each test case, the first line will have 3 ints, r, c, and l. The next r x l lines will be a 3 dimensional maze, whose dimensions are l levels by r rows tall and c columns wide. Somewhere in the maze there will be a character S, the starting position, and E, the end position. You cannot move diagonally.

**Output**

Print out the maze number in the format “Rescue #mazenumber: ”. If it cannot be solved, print out NO. Otherwise, print out the number of steps it takes to get to the end. Each maze will have one line of output.

**Constraints**

1 ≤ r ≤ 20

1 ≤ c ≤ 20

1 ≤ l ≤ 20

**Example Input File**

3

4 4 4

**Example Input File cont.**

.#..

.##.

.S#.

5 4 3

#..#

..##

.##.

.#.#

...E

#..#

#.##

#...

#..#

##.#

..##

.#.#

.#..

.##.

.S#.

**Example Output to Screen**

Rescue #1: 9

Rescue #2: NO

Rescue #3: 14

S...

#..#

##.#

....

####

####

####

###.

####

####

####

###.

####

####

####

###E

5 4 3

#..#

..##

.##.

.#.#

..#E

#..#

#.##

#...

#..#

##.#

..##

.#.#

**18. Flow**

**Program Name: Flow.java Input File: flow.dat**

Andrew has recently got into the IPhone app “Flow”. In flow, there is a square grid populated with empty cells and colored dots in pairs of two. Empty cells are denoted by a period, and dots are denoted by lowercase letters. The goal of flow is to connect each dot with its corresponding part. Two color’s paths may not cross though, so every empty cell can only be occupied by one color. Given a flow grid, connect all dots to their respective partner and print out the completed flow grid. A completed flow grid will have no empty cells left.

**Input**

The first line of input will contain a single integer *n* indicating the number of test cases to follow.

Each test case will consist of:

1. A line containing a single integer, *m*, indicating the size of the grid
2. *m* lines representing the unsolved flow grid

**Output**

For each test case, output the solved flow grid

Include a space between test cases and at the end of the output.

**Constraints**

2 < m < 10

There will be at most 9 different pairs of dots

For each grid there will be one and only one solution

**Example Input File**

2

**Example Output To Screen**

yyyyy

bbbby

bggby

bgrby

rrrbb

yyyyyyyyy

yrlbbbbby

yrlloooby

yrrrrpppy

ygggrpyyy

ypppppymu

uuyyyyymu

ummmmmmmu

uuuuuuuuu

5

y....

.....

..g..

bgr.y

r...b

9

.........

.rlb.....

...lo.ob.

.......p.

.g.gr....

yp.....mu

.uy......

.m.......

.........