Parallel Processing Projects 2017/2018

General Note: you have to implement the project in both sequential and parallel code

1. ***Sudoku Solver***

Place a digit from 1 to 9 into each of the empty squares so that each digit appears exactly once in each of the rows, columns and the nine outlined 3x3 regions.

Sudoku

		2	7		6	1		
			3		1			
4								5
1	9						5	8
7	3						9	1
2								6
			1		3			
		7	4		5	8		

Solution

9	8	2	7	5	6	1	4	3
6	7	5	3	4	1	9	8	2
4	1	3	2	9	8	6	7	5
1	9	4	6	3	2	7	5	8
5	2	8	9	1	7	3	6	4
7	3	6	5	8	4	2	9	1
2	5	1	8	7	9	4	3	6
8	4	9	1	6	3	5	2	7
3	6	7	4	2	5	8	1	9

To give you 9 * 9 grid, in this grid has been filled in part of the yard numbers. Now is this game will have any solution? No solution output of NO, or output YES.

Input

The first line contains the integer N ($1 \leq N \leq 100$) representing the number of test cases.

Each test case contains 10 lines:

First 9 lines: each line contains 9 characters. The meanings of valid characters: -- '1' - '9' : the number of this grid. -- '?' : Blank Grid.

The last line is a blank line.

Output

For each test case, you are to determine whether or not it has solution. Print the solution if it has, and "NO" if not.

2. Maze

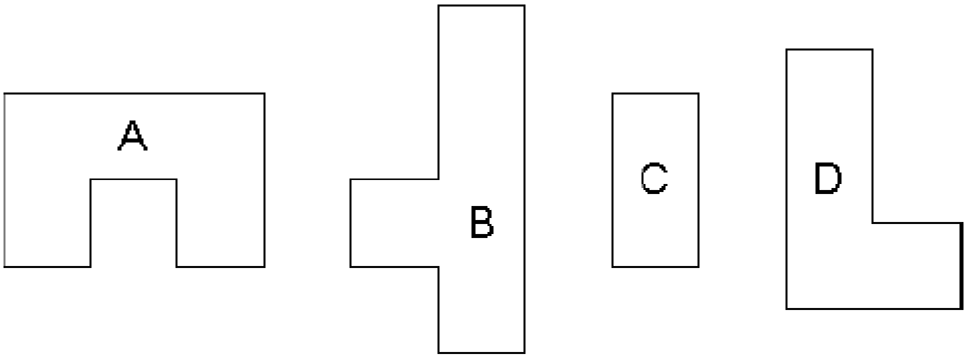
You’ve landed on an alien planet, in the middle of some strange maze, and you want to find the quickest way out.

The maze can be thought of as a m×n grid where some of the squares in the grid are blocked. You know the entire maze (including the size of the grid, which squares are blocked, and your current location). You can drive north, south, west, or east from each square in your hovercar, but not diagonally. Unfortunately, your hovercar was damaged during the rough landing onto the planet, and it can’t turn left or make U-turns: the hovercar can only go straight or turn right. For instance, if you leave the current square by going north, you’ll only be able to leave the next square by going north or east; west is not possible, because that would require a left turn, and south is not possible either, because would require a U-turn. Your hovercar was landed facing north, so on your first move you will need to drive north.

Find an efficient algorithm to find the shortest route to escape from the maze and implement it.

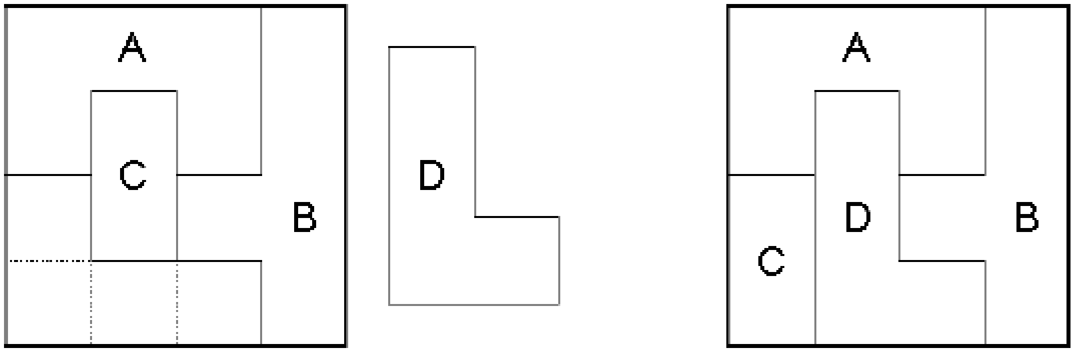
3. Make a square

Make a square with size 5X5 by using 4 or 5 pieces. The pieces can be rotated or flipped and all pieces should be used to form a square. Example set of pieces.



There may be more than one possible solution for a set of pieces, and not every arrangement will work even with a set for which a solution can be found. Examples using the above set of pieces…

Rotate piece D 90 degree then flip horizontal {R 90 + F H}



Input:

The first line contains number of pieces. Each piece is then specified by listing a single line with two integers, the number of rows and columns in the piece, followed by one or more lines which specify the shape of the piece. The shape specification consists of 0 or 1characters, with the 1 character indicating the solid shape of the puzzle (the 0 characters are merely placeholders). For example, piece A above would be specified as follows:

```
2 3
111
101
```

Output:

Your program should report all solution, in the format shown by the examples below. A 4-row by 4-column square should be created, with each piece occupying its location in the solution. The solid portions of piece #1 should be replaced with `1' characters, of piece #2 with `2' characters. Last line displays the pieces which are changed in the original form. If the piece rotate with angle 90{R 90}, and flip vertical or horizontal {F V or F H}. For cases which have no possible solution simply report "No solution possible".