Practice Problem #1: Tic-Tac-Toe game engine

Create a Tic-Tac-Toe game engine with the following features –

1. It should take inputs from the Standard input stream (keyboard),
2. It should end when the Tic-Tac-Toe is won by one of the players, or there is a draw, and display the message accordingly.

	1	2	3
1	X	O	X
2	O	X	
3	O		X

Player 1 wins.

For 1st part:

You will take input from the keyboard. Each input will consist of two space-separated integers in the range of [1,3]. E.g. input “3 3” refers to the bottom-right cell. The players will input the values turn by turn. “X” is associated with the cells chosen by player #1 and “Y” is associated with the cells chosen by player #2.

For 2nd part:

As soon as one of the player wins, display the message accordingly and end the program.

Bonus:

Validate the inputs for correctness. If a player enters the inputs that are outside the range, or present in the previous inputs, display error message and re-take the input from the player.

Example input:

1 1
1 2
1 3
2 1
2 2
3 1
3 3

Example output:

Player #1 wins

Practice Problem #2: Turtle Graphics Sketchpad

Being a PC user, you must have encountered the logo language before. It made the concept of *turtle graphics* popular. Imagine a turtle under your control moving around a room. It holds a pen in one of the two positions – **up** or **down**. When the pen is down, the turtle traces out shapes as it moves; while the pen is up, the turtle moves freely without writing anything. You will simulate the operation of the turtle to create a computerized sketchpad by exploiting the power of 2-D arrays.

(**Hint:** How to sketch? Take an 80x80 array as the sketchpad interface. Fill each element with a ' ' (ASCII 32) character. Place the turtle in the centre, with pen up by default and facing right direction (>). Keep track of the turtle's co-ordinates and replace ' ' with a '*' (ASCII 42) on a cell visited by turtle when the pen is down.)

Commands	Meaning
UP	Pen up
DOWN	Pen down
RIGHT	Turn right by 90 degrees
LEFT	Turn left by 90 degrees
FORWARD X	Move forward by X steps (where, X is an integer and 0 < X <= 10)
SHOW	Print the sketchpad
END	End of data

Input Format:

The input will follow the format shown in the above table's **Commands** column. Take input from the standard input stream.

Output Format:

Output the 80x80 array for each input line containing `SHOW`.

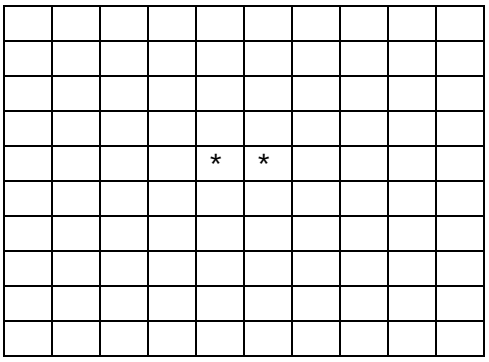
Example Input:

For the sake of demonstration, I am using a 10x10 grid (instead of the actual 80x80 used in program).

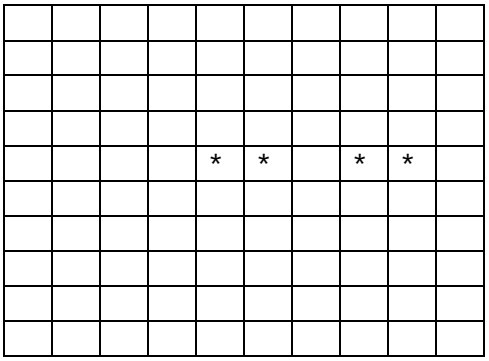
```
DOWN
FORWARD 2
RIGHT
SHOW
LEFT
UP
FORWARD 1
DOWN
FORWARD 2
SHOW
END
```

Example Output:

(The borders are only for clarity. Print the 80x80 sketchpad with each cell either ‘ ‘ or ‘*’)



Explanation: The turtle starts at the centre (row, col) = (5,5) and is facing EAST. It is asked to put the pen down and move forward by two steps, thus painting (5,5) and (5,6). When moving forward with pen down, the turtle paints the cell it was previously on. The turtle is now at (5,7). It turns right, thus facing SOUTH.



Explanation: The turtle shifts left, thus facing EAST and still at (5,7). Since the pen is up and turtle moves one step forward, there is nothing sketched at (row, col) = (5,7). The turtle puts the pen down again, and moves two steps forward, thus painting (5,8) and (5,9).

TEST INPUT:

DOWN
FORWARD 10
RIGHT
FORWARD 10
RIGHT
FORWARD 10
RIGHT
FORWARD 10
SHOW
END

[*EXPECTED OUTPUT:* This should draw a square with side length 10, with the top-left vertex at (40,40).]