Two Dimensional Array Practice Problems

1) Write a static method that takes in a two dimensional array and find the value of the largest number stored in that array. Ask the user to enter the number of rows and columns for the array in main, then read in the values into the array. Then, call the static method that finds the maximum value, store this in main and print it. Here is the method prototype:

```
public static int maxVal(int[][] nums);
```

2) Write a static method that takes in a two dimensional array and find the maximum of all the row sums. A row sum is simply the sum of each number on that row. Ask the user to enter the number of rows and columns for the array in main, then read in the values into the array. Then, call the static method that finds the maximum row sum, store this in main and print it. Here is the method prototype:

```
public static int maxRowSum(int[][] nums);
```

Feel free to write any helper methods you would like to write.

3) Write a method that takes in a two dimensional integer array and a String containing the characters "R" and "D" only. The String represents moves taken by a person from the entry in row 0, column 0 of the integer array. "R" represents moving right, which adds 1 to the column of the location and "D" represents moving down, which adds 1 to the row of the location. The total score of a path is the sum of the values of each array cell visited on that path. For example, for the array shown below and the String "RDRRDRR"

3	8	2	9	1	6
12	1	15	3	8	6
7	8	4	9	11	2

The score is 3 + 8 + 1 + 15 + 3 + 9 + 11 + 2 = 52.

If the directions take you off of the grid (out of bounds), then have the method return -1. Here is the method signature:

```
public static int pathSum(int[][] grid, String moves);
```

4) Write a program that creates a 5 x 5 game board and fills it with random integers in between 0 and 4. When displayed, the user will only see underscores. The user starts at the top left corner of the board (row 0, column 0) and their goal is to get to the bottom right corner of the board(row 4, column 4). At each step, ask the user if they want to go up, down, left or right. Assume the user enters a valid move. On a typical move, the number on the square to which the user moves is the amount added to her score. If the user moves to a square storing a 4, the game ends and the user loses (with a score of 0). If the user moves to a square storing a 0, the game ends and the user's score is equal to however many points the user had acquired up until that point. Alternatively, if the user makes it all the way to (4,4) without the game ending, the user's score is however many points she acquired on the full journey, including the points earned for the last square. (Note: If this last square stores a 4, the user gets these points added to their score.)

5) Add error checking for the program from #3 so that if the user enters a direction that moves him off the board or to a square he's previously been to, the move is not allowed and the user is asked to re-enter. As an added bonus, when prompting the user to move, only present them with a list of valid moves out of the four possible moves.

6) A Latin Square is an n by n square where each row contains the integers 1 through n, exactly once and each column contains the integers 1 through n, exactly once. For this problem, we define a Row Latin Square to simply be an n by n square where each row contains the integers 1 through n, exactly once. It's not necessary for the rows themselves to be unique. Write a method that takes a two dimensional array guaranteed to have the same number of rows and columns and returns true if the square is a valid Row Latin Square and false otherwise. Note: it's possible that entries in the input square are NOT in between 1 and n, inclusive. Here is the method signature:

```
public static boolean isRowLatinSquare(int[][] square);
```