Week 09 Multi-dimensional Arrays

Part 1 Matrix

In this part, we can make it easier to assume that all of the matrix entries are integers.

Task 1

Write a function `transpose(m)` to take in a matrix `m` with dimensions `r` x `c`, and return the transpose of `m`, that is a `c` x `r` matrix.

```
>>> pprint(m)
[[1, 1, 1, 0, 1, 0, 0, 1, 0, 1],
 [1, 0, 1, 0, 0, 0, 1, 0, 0, 0],
 [0, 0, 1, 1, 0, 0, 0, 1, 1, 0],
 [0, 1, 1, 1, 1, 0, 0, 0, 1, 1]]
>>> pprint(transpose(m))
[[1, 1, 0, 0],
 [1, 0, 0, 1],
 [1, 1, 1, 1],
 [0, 0, 1, 1],
 [1, 0, 0, 1],
 [0, 0, 0, 0],
 [0, 1, 0, 0],
 [1, 0, 1, 0],
 [0, 0, 1, 1],
 [1, 0, 0, 1]]
```

Task 2

Write a function to multiply two matrices.

```
>>> m1 = [[1,2,3],[5,6,7],[9,10,11],[13,14,15]]
>>> m2 = [[4,3,2,1,8,1],[1,2,3,4,3,1],[5,6,7,8,1,2]]
>>> pprint(matMul(m1,m2))
[[21, 25, 29, 33, 17, 9],
 [61, 69, 77, 85, 65, 25],
 [101, 113, 125, 137, 113, 41],
 [141, 157, 173, 189, 161, 57]]
```

Task 3

Write a function `minorMatrix(m,i,j)` to find the minor matrix of `m` without row `i` and column `j`.

```
>>> pprint(m)
[[21, 25, 29, 33, 17, 9],
 [61, 69, 77, 85, 65, 25],
 [101, 113, 125, 137, 113, 41],
 [141, 157, 173, 189, 161, 57]]
>>> pprint(minorMatrix(m,2,3))
[[21, 25, 29, 17, 9], [61, 69, 77, 65, 25], [141, 157, 173, 161, 57]]
```

Task 4

Write a function det(m) to compute the determinant of m

```
>>> m = [[6,1,1],[4,-2,5],[2,8,7]]
>>> det(m)
-306
>>> m = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
>>> det(m)
0
```

Part 2 Maze

Let's assume a maze is an r x c grid. And a 0 in the grid represents an empty space and a 1 represents a blocked space.

Task 1 Generate a random maze

Write a function createRandomMaze(n,m) to generate a random maze with size of n rows by m columns with an equal random chance for empty and blocked spaces.

```
>>> maze = createRandomMaze(8,14)
>>> mTightPrint(maze)
01111111010011
00000001000000
10000100100000
10110010000010
00001100011001
10000101010100
10001001011000
10010100110100
```

Task 2

Given that you can enter from the position `(0,0)` and the exit is at the lower right corner of `(n-1,m-1)`. Write a function `isSolvableMaze(m)` to take in a maze `m` and return `True` if you can go from the entrance to the exit, or `False` otherwise. E.g. the maze above is solvable.

Extra Tasks

- Write some code to find a maze generated by the first task that have a size of 15 x 30 and solvable.
- Other than `mTightPrint()`, write your own function to *beautify* your maze presentation.
- When you find out a maze is solvable, leave a trace, e.g. put a '.' in the matrix to show the trail