Week 6 Multi-dimensional Arrays

Part 1 Matrix Operations

In this part, you can assume that all matrix entries are integers.

Task 1 Transpose

Write a function `transpose(m)` that takes in a matrix `m` with dimensions `r` x `c` and returns the transpose of `m` which has dimensions `c` x `r`.

```
>>> 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 Multiplication

Write a function `matMul(m1,m2)` that returns the result of multiplying matrix `m1` by `m2`.

```
>>> 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 Minor Matrix

Write a function `minorMatrix(m,i,j)` that returns 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 Determinant

Write a function `det(m)` that returns the determinant of matrix `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 that consists of 0s and 1s, where 0 represents an empty space and 1 represents a blocked space.

Task 1 Random Maze

Write a function `createRandomMaze(r,c)` which returns a maze of size `r` x `c` where every space has an equal random chance of being empty or blocked.

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

Task 2 Solvable

Suppose you can enter a `r` x `c` maze from its top left corner at position `(0,0)` and exit from its lower right corner at position `(r-1,c-1)`. Write a function `isSolvableMaze(m)` that takes in a maze `m` and returns `True` if you can eventually reach the exit from the entrance, and `False` otherwise. In other words, determine if the maze is solvable. (You can only move horizontally or vertically, not diagonally.)

Extra Tasks

- Write some code to find a maze generated by Task 1 that has a size of 15 x 30 and is also 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. replace the space with a '.') to show the path found.