Lab 10

Exercise 1: Write a function `fromListToMatrix`(A) which takes an adjacency list representation of a graph A and outputs an adjacency matrix representation of the same graph. Similarly write a function `fromMatrixToList`(A) which goes in the other direction.

Exercise 2: Write a function `connected`(A, i, j) which takes an adjacency list A and two vertices i,j, and outputs `True` if there is a path connecting i to j in the graph represented by A, and outputs `False` otherwise.

Exercise 3: Write a function `path`(A, i, j) which takes an adjacency list A and two vertices i,j, and outputs a path from i to j in the graph represented by A in `list` form, or [] if no such path exists. For example, if we can get from vertex 0 to 5 by taking the sequence of edges $(0, 1), (1, 4), (4, 7), (7, 5)$ in some graph, then the function should return the `list` [0, 1, 4, 7, 5].

Exercise 4: Write a function `numComponents`(A) which takes an adjacency list A and outputs the number of connected components in the graph represented by A.

Exercise 5: Write a function `distance`(A, i, j) which takes an adjacency list A and two vertices i,j, and outputs the length of the shortest path from i to j in the graph represented by A. This can be done by modifying the code for bfs to keep track of distances.