NAMES: John Karuntzos / Kristopher Hoce / Dakota Krogmeier

Worksheet 22: Work with either ONE or TWO other students to complete this worksheet. You must work with at least one other student on this assignment, and should submit one paper.

1. Develop an efficient algorithm to add an edge to an adjacency list representation of a directed graph (3 points):

   KEY IDEA: To have three checks on whether a vertex exists in a graph and whether it was connecting to itself. The first check can be done with to determine if either of the integer values are greater than the number of vertices in the graph. Which if they are, they will return that the vertex does not exist. The second check checks if the integers of x and y are equal to each other then that returns that they are the same vertex if the values are equal. The last check is to see if the edge is already in the graph. After going through the 3 checks then connect the vertices of the vertex x and y by adding a new edge.

   Algorithm AddEdge(G, (x,y)):

   Input: A directed graph, G = (V, E), represented as an adjacency list, and an edge, (x, y)

   Output: The directed graph, G = (V, E ∪ { (x, y) }), represented as an adjacency list

   PROCESS:

```
if   x >= n  OR  y > n
       print("not in graph")


if  x == y :
       print("Same Vertex")

if  adj[x] contains y:
       print("edge already exists")

else :
       adj[x].add(y)
```

2. Develop an efficient algorithm to remove an edge to an adjacency list representation of a directed graph (3 points):

KEY IDEA: To have three checks on whether a vertex exists in a graph and whether it was connecting to itself. The first check can be done with to determine if either of the integer values are greater than the number of vertices in the graph. Which if they are, they will return that the vertex does not exist. The second check checks if the integers of x and y are equal to each other than that returns that they are the same vertex if the values are equal. The last check is to see if the edge is not in the graph. After going through the 3 checks then remove the edge between the two designated vertices on the graph.

Algorithm RemoveEdge(G, (x,y)):

Input: A directed graph, G = (V, E), represented as an adjacency list, and an edge, (x, y)

Output: The directed graph, G = (V, E \ { (x, y) }), represented as an adjacency list

PROCESS:

```
if   x >= n  OR  y > n
       print("not in graph")


if  x == y :
       print("Same Vertex")


if adj[x] does not contain y:
       print("No edge exists")

else :
       adj[x].remove(y)
```