NAMES: John Karuntzos / Kristopher Hoce / Dakota K

Worksheet 17: 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.

Realizing that Dijkstra's algorithm finds the shortest paths FROM a **source** vertex to every other vertex in the graph, modify this algorithm to develop an algorithm to find the shortest paths TO a given vertex, called the **destination** vertex. (6 points)

KEY IDEA: Starting with the destination vertex, working towards all of the other nodes in the graph, g. Using an idea similar to Dijkstra's algorithm, finding the shortest path from the destination node to all other nodes. Where the destination node is the starting point and to leave to go to the next node there is a check to find which length is the shortest. After finding the shortest length, go to the next node. After traveling to the next node record the length of that value plus the value of the length it took to get to that node, if any, in the output array. While the new node checks the length to the nodes it connects to, record those values as the path length to those nodes then return to the destination node. This repeats throughout the graph where as different node paths from the destination node might have shorter travel lengths the ones recorded earlier on, which will be taken care with a comparison check for the node whenever traveled to it. If the value is shorter take the shorter length and throw the old one out. After going through the entire graph, return the array of lengths

Algorithm DijkstraDestination(G, f):

Input: A graph G = (v,e) , a destination vertex f

Output: Array of the shortest paths from each vertex to the destination

PROCESS:

```
distances <— [infinity,…]  //Length of V
sptSet  <— [False,…..]      //Length of V

distances[0] <— 0

for count <— 0 to V-1:
    u <— minDist(distances, sptSet)
    sptSet[u] <— true

    for v <— 0 to V:

        if sptSet[v] = false:
            if distances[u] != infinity:
                if distances[u] + G(u,v)  <  distances[u]:
                    distances[u] <— distances[u] + G(u,v)


Return distances
```

```
minDist( distances, sptSet):

    min <— infinity
    minIndex <— -1
    for v <— 0 to V:
        if sptSet[v] = false:
            if distances[v] <= min:
                min <— distances[v]
                minIndex <— v

    return minIndex
```