Net ID: wpicc001
Name: William Patrick Picca

Sources Used:
Communicated with fellow student: Amneh Alsuqi

Problem A: Single Source Shortest Path
Submission ID: 136695922

For the first programming assignment I started off by taking the input for the total number of nodes followed by the number of edges. I then take in the start and end target node, to then take in all the following edges and their edge costs into a vector. With the vector of edges obtained, I will call my shortestPath function

This function works by creating a priority queue from the passed in vector edges, in addition to creating a new vector to hold in and update the node distances as we progress. Given that we can start at any node, I set the distance of the starting node to 0, the starting node being the first thing pushed into the queue.

Then, for as long as that queue is not empty, I obtain the connected node and all other nodes directly adjacent. From there the cost of the edge of those 2 nodes is collected from the vector of edges. If the distance of an adjacent node happens to be greater than the current node in summation with the weight of the connecting edge, this means I found a lower cost path. I push that node into the queue, and the process repeats for any remaining adjacent nodes.

When no more adjacent nodes remain, we repeat this whole process with the next node in the queue, which happens to be the one with the lowest costs since it's a priority queue. At the end, we now have the SSSP values for all nodes in relation to our starting node, so we print out the cost for our target node, which is our final SSSP answer.

Problem B: Minimum Spanning Tree
Submission ID: 136804844

For the second programming assignment it starts very similar to the first. The program takes in the inputs for the number of nodes, number of edges, and then the start and end node of an edge followed by its edge weight. The edge is then shoved into a vector and the vector is passed into the MST function.

Within the MST function I create a priority queue just like in problem A, in addition to a vector of booleans to represent whether or not a given node has already been visited. At this point we start at node 0, whereas the distance is set to 0 for itself.

Just like before we begin processing the queue, starting at node 0, and obtaining all adjacent nodes. If any of these nodes have not already been visited, and the weight of that edge is less than the current noted weight of that node, we update that node's weight with the weight of the edge, in addition to pushing this new node into the queue.

This whole process repeats for all adjacent nodes, until no more nodes are left not visited in the tree. At the end we then sum up all the noted distances for all the nodes, and return it as our smallest cost final answer.

Problem C
Submission ID:

For the 3th programming assignment

Problem D:Add Oil
Submission ID: 136805355

For the 4th programming assignment not much else must be needed to be changed to accomplish this. To be brief, I take the entirety of the problem 2 code, but instead of summing up all our distances at the end of the MST function, I instead take the max of them. The returned result would be the lowest fuel tank capacity needed to visit all nodes.