1. **Design**
    a. The problem that I attempted to solve was to find the shortest path from Sweet Home Oregon to 6 other points in Oregon Brownsville, Lebanon, I-5, Tangent, Albany, and Corvallis. The PDF that I turned in to GitHub has the remaineder of my drawn-out Design I provide a design for Dijkstra's algorithm and for Prim's Algorithm
2. **Test**
    a. In my design write up, I have provided 5 visual tests for the shortestPath() function and minimumSpanningTree() by drawing the graphs and outlining the outputs for the shortest paths and all the minimum spanning trees to demonstrate that the output of the program is what is expected.
    b. In Function Tests() in file Graph.cpp lines 266 -> 277 tests the addNode() Function
    c. In Function Tests() in file Graph.cpp line 272 is a visual test for the display() Function
    d. In Function Tests() in file Graph.cpp lines 279 -> 303 tests the connectNodes() function
    e. In Function Tests() in file Graph.cpp lines 305 -> 312 tests the getSize() function
3. **Implementation**
    a. Function addNode() in file Graph.cpp lines 34 -> 38 adds nodes to the graph
    b. Function connectNodes() in file Graph.cpp lines 40 -> 63 connects the nodes in the graph
    c. Function shortestPath() in file Graph.cpp lines 79 -> 152 finds the shortest path from a given node to all other nodes in the graph
    d. Function minimumSpanningTree() in Graph.cpp lines 165 -> 251 find the shortest path to reach all other nodes in the graph
4. **Complexity**
    a. For my shortestPath() Function ideally using Dijkstra's Algorithm my O value would be O(E+Vlog(V)) but mine is far from ideal and its big O value is O(V^2) due to the nested for loop in my while loop
    b. For my minimumSpanningTree() the worst case scenario would be O(V^2) due to my nested for loop in a for loop in a while loop
    c. addNode() functions time complexity is O(1) because you only ever add one node at a time
    d. connectNodes() time complexity is O(V) because of the one for loop to find the source and destination node
    e. getSize() is O(1) because it returns one node at a time
    f. display() is O(V) because of the for loop to display the node values
    g. printPath() is also O(V) because while it does not have a loop in it, it is recursively called by a for loop
    h. Lastly printSlash() has one for loop making it O(n)