Shaotai Hu DS210 Project Conclusion & Result Analysis:
https://snap.stanford.edu/data/roadNet-CA.html

(Note: the website says the data is undirected. But I downloaded the data, and it said it is directed in the data itself, and it makes sense for roads to be directed. Thus, I used a directed graph for my project)

Using California road network data set from Stanford Large Network Dataset Collection to create a directed graph, this projects aims to answer the question:
"Is California road network design logical and effective?"

To answer this question, I must first consider the aspects of logical and effective road network design. I will only consider aspects that can be answered using graphs and Rust. Thus, the aspects of logical and effective road network design this project will analyze are:

1. Downtown areas or city like areas should have various roads leading to them, and leaving from them. This means cars can reach these important city like areas from various directions, and leave in various directions. Such road design allows more cars to pass popular areas faster, and decrease traffic as a result.
2. The average distance it takes from any location to reach downtown/city like areas should be reasonable. Reasonable can be define by the overall average distance from any popular node to every other node should be equal to or more than the average distance from any node to get downtown/city like areas. If this is true, people are able to reach important & popular locations within reasonable time, no one is left too far from the cities of California.

- To run my program, simply use "cargo run --release" in terminal on VS Code, cargo run is too slow. Please clone/download my Github repository to your machine, and run with VS Code, as I am not sure if it will work properly when not using VS Code, since I made this in VS Code. Modules are in separate files. I have formatted my program that just using cargo run --release will print the supposed results in the terminal.

To answer the two central questions, I used average distances and centrality measures. Firstly, I must first find the nodes with the highest degree or most edges. The nodes/vertices are intersections. These nodes have more edges, meaning there are more roads leading to them. This could mean the node is an road intersection of great importance, as it likely acts as a connecting point for many roads. I found the top 50 nodes with highest degree by sorting and reversing a vector with each node's edge lengths in the first column. I also found the top 50 nodes' average distances to every other node, as this is used for comparison in the next section.

```
Top 50 nodes with highest degree
degree: 12, node: 562818, average distance to every other node: 305
degree: 10, node: 534751, average distance to every other node: 289
degree: 10, node: 521168, average distance to every other node: 272
degree: 9, node: 1795416, average distance to every other node: 263
degree: 8, node: 1872330, average distance to every other node: 305
degree: 8, node: 1780830, average distance to every other node: 261
degree: 8, node: 1631015, average distance to every other node: 280
degree: 8, node: 1616784, average distance to every other node: 288
degree: 8, node: 1545464, average distance to every other node: 244
degree: 8, node: 1495419, average distance to every other node: 264
degree: 8, node: 1484788, average distance to every other node: 269
degree: 8, node: 1389459, average distance to every other node: 318
degree: 8, node: 1275439, average distance to every other node: 322
degree: 8, node: 1272137, average distance to every other node: 321
```

From the above result, we see that there are 12 edges to the most center node, meaning that this intersection is important and well designed. Other important intersections have from 11-7 edges, which are all indications of logical and effective road network design. Most edges mean faster passing and less traffic. Then I found overall average distance from a node to every other node on the graph. Because the data set has 1,971,281 nodes, so I decided to randomly select 2000 nodes that, since randomly selected without bias, is representative of the population. I then used breadth first search to find the average distance of each of the 2000 nodes to get to every other node. Now it

does not make sense to analyze all 2000 nodes' average distance. Thus, I summed the 2000 values, and averaged it to get a number that is representative of the overall average distance from a random node to every other node in the graph. Due to random selection, this value ranges from 303 to 308, and the average out of 10 runs is 305.

```
    Finished release [optimized] target(s) in 0.50s
     Running `target/release/shhu_project_git`
The overall average distance from a node to every other node is 304
```

When comparing 305, which is the overall average distance, to the top 50 nodes, we see that 64% of the top 50 nodes' average distance to every other node is shorter than 305, with the most center node at 305.

```
percent of top 50 center nodes that has lower /
                average distance than overall average distance is 64.0 %
```

This is an indication that more than half of the top 50 most center/important road intersections can be reached within a distance that is shorter than the overall average distance from a node to every other node. 64% indicates that the overall California road network design is fairly logical and effective. I use "fairly" because if the percentage was in the upper 80%, then this would be a more firm statement.

2 tests all passed

- Test 1 is going to see if average_distance() gives a distance within range of all possible average distances.
- Test 2 tests to see if the highest centrality degree from centrality() matches the max value in a vector of all number of edges in the graph.

```
    Finished release [optimized] target(s) in 0.79s
     Running unittests src/main.rs (target/release/deps/shhu_project_git-1d662d8ccc157a71)

running 2 tests
test test_centrality ... ok
test test_distance has been running for over 60 seconds
test test_distance ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 127.76s
```