CIS7 Project Documentation

1. Team name: CAR
   Members: Ivan Sebastian Perez, Eric Coria, Marlon Jimenez
2. Project Information and details:
   - What problems are you solving in this project?
     This project aims to assist in trip planning to suit various needs of an Inland Empire salesman. The program will allow the user to select between viewing possible routes, cheapest route, shortest round trips, and a general map to help them plan their trips throughout the Inland Empire.
   - What solutions are you implementing in the project?
     We implemented multiple functions to attempt to decrease the amount of space taken up by the program without sacrificing the performance of the program. When finding the possible routes, we set the relationships of the cities in the main program and created two functions to find the possible routes and print the results to the screen. In addition to displaying the possible routes, an adjacency list, or a city's connecting cities, was also implemented into the program to help with trip planning and routing. Being able to display both the possible routes, and also a list of connecting cities is a solution that resolves the problem of the salesman not knowing which order, or which combination of routes to go in when route planning. The cheapest route can be determined by calculating the shortest path starting from Riverside and traveling through all the cities while traveling the shortest amount of distance. The map showed all the possible routes, similar to choice 1, but also included the distance between them. This would help the user visualize how far the cities were from each other and which route they could take.
   - Provide explanation of calculations and algorithm implementation.
     The possible routes were found by manually setting and entering the relationships of the cities (Edges) into the struct Edge and then passing that data to the graph class and to one of the printList functions to print the results to the screen. The cheapest route was found by creating relationships between the edges, the cities, and their line weight to determine which route would cost the least.The shortest round trips were found by creating an adjacency list. The adjacency list was implemented into the program by creating a data type that could store the city, and its three adjacent cities. Then, the struct was used to create a class that could store an array of these data types, then using functions, each member of this array would then be assigned a city and its adjacent cities and properly displayed to the user. The map was made by creating a matrix which took the relationship of the edges and their line weight to calculate the number of edges, pass the data to construct the graph, and print the adjacency

list representation of the graph and all neighboring vertices of the vertex. This data was passed on to an alternate printList function that organizes the information in a readable manner for the user.

- What are the program objectives? Explain how your program is interacting with the user and its purpose.
  The objectives of our program were to display the variations of trips between 4 cities starting from Riverside, and then also display the shortest route and the cheapest route that could be taken when considering what city to arrive at. Our program interacts with the user by asking the user to set a destination and then displaying the routes to said destination, the purpose being to act as a sort of GPS system to help the user be able to arrive at any destination taking any route possible with some recommendations for routes.
- How are discrete structures implemented in the C++ program?
  Our prompt was to be able to display the variations the solar salesman could take to arrive at each city, starting at Riverside, as well as determining the shortest route possible. The process of creating a program that could display these variations were made easy by using graphing techniques we learned in class, by using an adjacency list, and using an adjacency matrix, implementing our city map into our program and displaying the variations was a process made easily achievable, compared to trying to make the same kind of program without knowing these techniques before hand would have made the process much more difficult, or would take much more time to create a solution.
- What are the limitations of the program?
  One of the major limitations of our program is the lack of user input for making trips outside the Inland Empire, such as neighboring cities not too far away. The user can only see routes in the noted cities by the program and not anywhere else. If the demand warranted making this program more accessible outside the Inland Empire, these changes would help make the program accessible to a wider audience. Another limitation is the basic way the outputs are displayed, for example, a possible route is displayed as “1 -> 2 -> 4 -> 3.” The legend we created helps with making the output more readable but it is still clunky and unpolished for widespread use outside of our salesman friend. While there are several other aspects that can be improved upon from our program, we believe these are a few of the main issues to address.
- Provide recommendations on improving the limitations of the program.
  We recommend that future iterations include increased options for user input to select different cities, enter miles between the cities, and polishing the outputs for increased user experience. By adding user input for neighboring cities and their distance, this application can be used by neighboring salesmen to plan their routes depending on their needs. We also recommend displaying the name of

the cities alongside the numbers they are represented by to increase user accessibility. Example: “ 1) Riverside -> 2) Hemet -> 3) Moreno Valley -> 4) Perris”

3. Flowchart