Alex Negron
CS 201 Final Project Plan

1. User Interface.

The user interface will consist of a printed menu of options listed as items 1-7 contained within a main method in an application class.

Item 1 will create a new station and add it to the CTA system. This will prompt the user to enter the appropriate fields that define a station, then print the generated system's information to the console for verification. Note that this will have to not only add the station to the appropriate list of stations, but it will also have to shift the positions of all the other stations that share the same line as the added station.

Item 2 will delete a station from the CTA system. This will prompt the user for the station's name, look up that station in the system, and then remove it. After removal, the position values indicating the other stations on this line's positions on that line must also be modified appropriately.

Item 3 will modify a designated CTA station. This choice will prompt the user for the fields of the station they want to alter, then alter them. Note that alterations made to coordinate positions and line positions are not going to be permitted, since this doesn't make sense: it would imply that we can pick up a station and put it elsewhere.

Item 4 will prompt the user for the name of a station then search the CTA system for that station. Since some stations share the same name, when this option is selected, the program will have to return all of the stations bearing that name.

Item 5 will find the nearest station to a user-given location

Item 6 will generate a route between two stations given by the user. This will generate routes between any two stations in the system, whether or not they share a line. Ideally, this program will also construct these routes such that they stations are presented in the order by which a user would take them.

Item 7 will exit the menu and terminate the program.

2. Tasks.

The input file will be read through a scanner that will loop through each line of the .csv file, split at the commas, and store each line in a list of CTAStation objects.

The input file will be read via three methods that extract the file's data for all the stations presented, all the lines, and all the transfer stations. These three methods will then allow for the construction of a "CTA" object, which encapsulates all the information we want to know about a CTA system: the stations, lines, and transfers.

The classes required are the GeoLocation, CTAStation, CTARoute, CTALine, CTA, and CTA_Application classes. Note that the CTARoute and CTALine classes are similar, but I choose to distinguish between a route and a line for the sake of simplicity while programming. Since a route can be user-defined, and a line cannot be and is fixed, it makes sense to me to separate the two.

3. Test Plan.

Testing will be done as each method is written to ensure that errors do not compound. In particular, I'm anticipating the route making function to require a lot of work. This function will be tested during development, as will the rest.

Once the program is complete, I plan on testing each of the menu options to ensure the loops are looping as I want them to. For each menu item, inputs will be given that test all (or nearly all) possible cases. For route making, it will be important to test several different pairs of stations and make sure that the routes generated make sense and are in order (I don't know about this ordering business yet, but I'll give it my best shot).