Computer Science 3270
Project 1
C Programming Project
Rainfall Flood Simulation

Notes

Some parts of this project contain mathematical formulas, but nothing more than you have seen in Math 165 or before. There may be parts of this project description that are ambiguous (not intentionally), but it is your responsibility to clarify the program specifications provided in natural language. Please read the assignments early and ask questions in class. Questions that lead to further specification or changes in the project specification will be posted in Piazza or in the announcements section of the Canvas course.

Introduction

Map data and location data are utilized in various ways and are ubiquitous in mobile applications. In this project, map data will be used to find routes and other properties of the map data for Ames, Iowa. However, your program must operate and will be tested on data from other locations.

Data Format

Road data is stored in a file with two distinct sections: the Points of Interest section and the Road section. The first section is the Points of Interest, with the first line of the file giving the number of entries. This is followed by a Point of Interest (POI), one per line. Each line consists of comma-separated values as follows: An ID number of the POI, the name of the POI, the latitude of the POI, and the longitude of the POI. Note that Ames is at 42 degrees North latitude and 93 degrees West longitude. North and East latitudes and longitudes are represented by positive numbers, while negative numbers represent South and West. Thus, Ames is located at a longitude of -93 degrees.

The second part of the file represents roads over a region. The format is similar to the POI section, with a single line that gives the number of entries in the section. This is followed by entries that define roads in terms of from/to nodes, along with the distance of the road segment in meters, the latitude and longitude of the segment's start, and the name of the road. An example entry looks like the following:

```
161179915,161133478,50.19,42.02475,-93.6468,Union Drive
```

Note that some road lengths may contain NaN, indicating that there is no length associated with the road. For the purpose of any calculation, these are considered to be zero.

Project 1, Part a

The purpose of this part of the project is to create the infrastructure for the assignment and to do a simple C program that will get you familiar with the process. This part of the project is divided into several parts.

1. Remove or rename any previous Git repositories that you have created with the following name, and create a new repository named "**coms3270P1**" Also remember that part of your grade will depend on README and DEVELOPER files. Please see the syllabus.

2. Create at least 3 small data test files in the format described above that will help you debug your program. These test files should be reasonably small so that any computation your program does can be verified by hand. As an example, consider a test file with one long road with only one intersection and two points of interest along the road. In this case, the road is divided into four segments with three intersections. The road section of the data file would contain four lines.

3. Create a function named "validate" that reads a data file from standard input and determines if the data file is valid. (Note that this will be updated in other parts of this project.) For this part of the project, the following conditions give an invalid file.

    a. Latitude or Longitude out of legal range. For example, the longitude is greater than or equal to 360.
    b. There is an empty TO or FROM Id or the road name is empty.
    c. The Id numbers contain any other character other than the digits 0 through 9.
    d. If either of the numbers that give the number of entries is 0 or negative.
    e. If latitude and longitudes have characters that do not allow conversion to a number using scanf.
    f. If the POI name is empty.

    The function returns an integer of 0 if the file is valid, and otherwise returns the line number of the file for the first invalid entry. Place and save this function in a file named data.c.

4. Create a "main" function that calls the "validate" function and place this function in the file "mapper.c" The main function then outputs if the data file read is valid by printing the single word "VALID" to stdout, or the line number where the first invalid data was found.

5. Create a header file named "data.h" that creates a correct prototype for the function defined in 3 above. Be sure to include correct guards for the header file.

6. Create a makefile for this project so that when a user types make in the top directory for the repository, the executable program named “mapper” that if run, the main program in the “mapper.c” file is executed.

7. Create and store your project in the gitlab repository git.las.iastate.edu. When completed and your program is correct and documented, create a tag with the name “mapperAcomplete”

8. At least 10% of points will come from documentation and following the syllabus requirements for the project.