Pointy GIS Pointers and Dynamic Memory Allocation

This assignment requires modifying the design and implementation of the Classy GIS to incorporate C++ pointers and dynamic memory allocation.

A good design for the Classy GIS can easily be turned into a good design for this project. Assuming you have a good design for the previous project, you will modify the designs of several of the classes from that design design.

The primary changes from the previous project involve the use of pointers. Neighbor relationships will now be represented by having city objects store pointers to other city objects; those pointers will probably be encapsulated within a neighbor class. The city objects themselves will now be allocated dynamically, and the GIS will use an array of pointers to organize the city objects, rather than storing them directly as in the previous project.

Another change is that the city objects will be stored in ascending alphabetical order (based on both city name and state abbreviation), and that searches of the ordered list of cities must use the binary search algorithm for improved efficiency. Searches performed before the list is sorted will use the linear search algorithm.

A final change is that the program will receive the names of the input and log files as parameters from the command line. See the section on program invocation for details.

All of the previous limits, such as the maximum number of cities and neighbors still apply. There is no requirement that you allocate any of the various arrays dynamically, but you may do so if you wish.

When thinking about your design, it is important to assign each of your classes the correct responsibilities. Give careful thought to the importance of information hiding, especially with respect to the relationship between the controller code and the GIS class. (Yes, that's a strong hint.)

Input Data

There is no change to the input data specification given for the Classy GIS project.

System Initialization

System initialization proceeds as in the Classy GIS project, with one change. After reading the list of cities, and storing them in the order they are given in the data file, you will sort the list, using the bubble sort algorithm.

Script File

The script file will have the same format as before. However, now when a new city is added to the database, it will be added at the proper location within the list, and when a city is deleted from the database, it will be physically removed and the array will be shorted accordingly.

Log

Everything said about the log output in the previous project specifications still applies. The log file header must echo the names of the input and log files that were used. In addition all searches must be instrumented, by displaying the name and state abbreviation of each city record that is examined during the search.

Since this project is not autograded, the precise formatting of the log file is up to you. In order to make it easier for the TAs to examine your log output, you must also number the commands in your output, starting the numbering at 1.

Implementation Constraints

This specification imposes a number of limitations on how you are allowed to implement your solution. Some of these constraints may appear to be unjustified. They are not. Regardless, failure to conform to them will certainly result in a grade penalty on the project.

- You are forbidden to use anything that is not ISO Standard C++. The course notes present only Standard C++. If you are not sure about something, ask. In particular, you may not use any of the non-Standard abominations commonly presented in the AP Computer Science course. These include, but are not necessarily limited to, `apstring` and `apvector`.
- You must use pointers as described above.
- You are explicitly required to use any dynamic allocation in this assignment (e.g., `new`, `delete`), as described above. You are required to make sure that any memory you allocate dynamically is also deallocated by your program.
- You are forbidden to use any STL containers, except `string`, on this assignment. In view of the restrictions given above, you are thus limited to using statically-allocated arrays for storing lists. This will impose some functional limitations on your solution; that's OK.
- You are required to write classes. You are explicitly forbidden to use `struct` types.
- You must implement your design using separate compilation. In particular, you must have a separate header file and cpp file for each class you write, and at least one cpp file for the driver code that uses those classes.
- Without exception, all class data members must be private.

This project will not be autograded; instead, you will schedule a demo with a TA. Because of that, some of the restrictions that were imposed on the earlier projects can be dropped. In particular, you may store neighbors in any order you like.

Program Invocation

This program will take the names of the input files and the log file from the command line. Assuming your executable is named `gis.exe`, you would type the following command to run it:

```
gis <city data file name> <route data file name> <script name> <log name>
```

The use of command-line arguments is illustrated in an appendix to the course notes, and will be discussed in class.

Submitting Your Program

Submit all the source files, and the `vcproj` file, for your implementation to the Curator in a single zip archive file. Do not submit unnecessary files! A penalty may be assessed if your zip file is unnecessarily large. You will be allowed to make up to three submissions to the Curator; it will be your choice at your demo as to which will be evaluated.

The *Student Guide* and submission link can be found at: http://www.cs.vt.edu/curator/

Evaluation

Shortly before the due date for the project, I will announce which TA will be grading your project. You will schedule a demo with your assigned TA. The procedure for scheduling your demo will be announced later. At the demo, you will perform a build, and run your program on the demo test data, which we will provide to the TAs. The TA will evaluate the correctness of your results. In addition, the TA will evaluate your project for good internal documentation and software engineering practice.

Note that the evaluation of your project will depend substantially on the quality of your design and documentation.

Pledge

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the pledge statement (provided on the course website) in the header comment for your program.

Failure to include the pledge statement may result in a substantial grade penalty.