San José State University
Department of Computer Engineering

CMPE 180A
Data Structures and Algorithms in C++

Fall 2020
Instructor: Ron Mak

Assignment #7

Assigned:	Tuesday, October 6
Due:	Tuesday, October 20 at 5:30 PM
CodeCheck:	http://codecheck.it/files/180312045951nql652nne06s1epyi0fdgpl
Canvas:	Assignment #7. U.S. Maps
Points:	130

U.S. maps
This assignment will give you practice writing a C++ application with a linked list and multiple classes. You will also further practice reading CSV files.

The output of your program will be two printed maps: an outline of the continental United States and the same outline with major cities superimposed on it.

You will complete four classes, **`Coordinate`**, **`City`**, **`Node`**, and **`SortedLinkedList`**. You are provided the complete source for main application, **`MapMaker.cpp`**.

Input files
Your program will read two CSV (comma-separated value) files as input data.

Input file **`boundary-data.csv`** contains the geographic coordinates of locations along the border of the U.S. Each line of the file contains one coordinate consisting of two double values separated by a comma:

latitude,*longitude*

Some example lines from the file:

```
34.299084,-119.362617
29.47158,-83.302581
28.707351,-82.653682
32.342609,-114.426405
34.044824,-118.532846
```

The lines are not sorted in any particular order.

Input file **`city-data.csv`** contains the names and states of major U.S. cities and their geographic coordinates. Each line of the file contains a city name, state abbreviation, and the latitude and longitude of the city's geographic coordinates, all separated by commas:

name,state,latitude,longitude

A few lines from the file:

```
San Francisco,CA,37.775,-122.4183333
San Jose,CA,37.3394444,-121.8938889
Santa Fe,NM,35.6869444,-105.9372222
Savannah,GA,32.0833333,-81.1
Seattle,WA,47.6063889,-122.3308333
```

The lines are sorted in alphabetical order by city name.

Input file **`boundary-data.csv`**:
http://www.cs.sjsu.edu/~mak/CMPE180A/assignments/7/boundary-data.csv

Input file **`city-data.csv`**:
http://www.cs.sjsu.edu/~mak/CMPE180A/assignments/7/city-data.csv

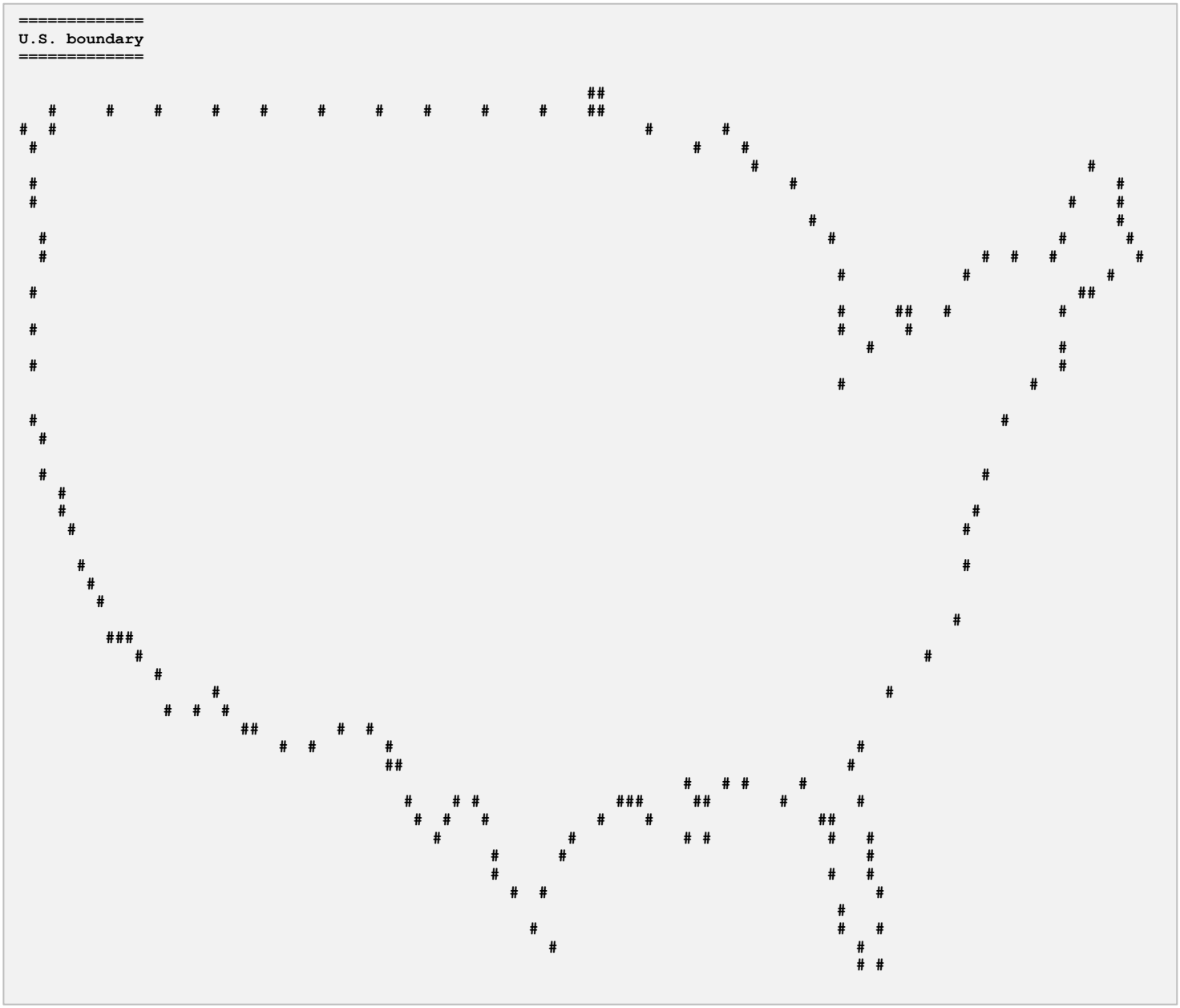
These input files are already uploaded to CodeCheck.

Reading comma-separated values
Your program must overload the input stream extraction operator **>>** to read the boundary and city CSV files.

To read the comma-separated values, you can use the **`getline`** function. It has an optional parameter to specify a delimiter character, such as the comma. See http://www.cplusplus.com/reference/string/string/getline/

U.S. boundary map

Your program must first read input file **`border-data.csv`** to print this boundary map:

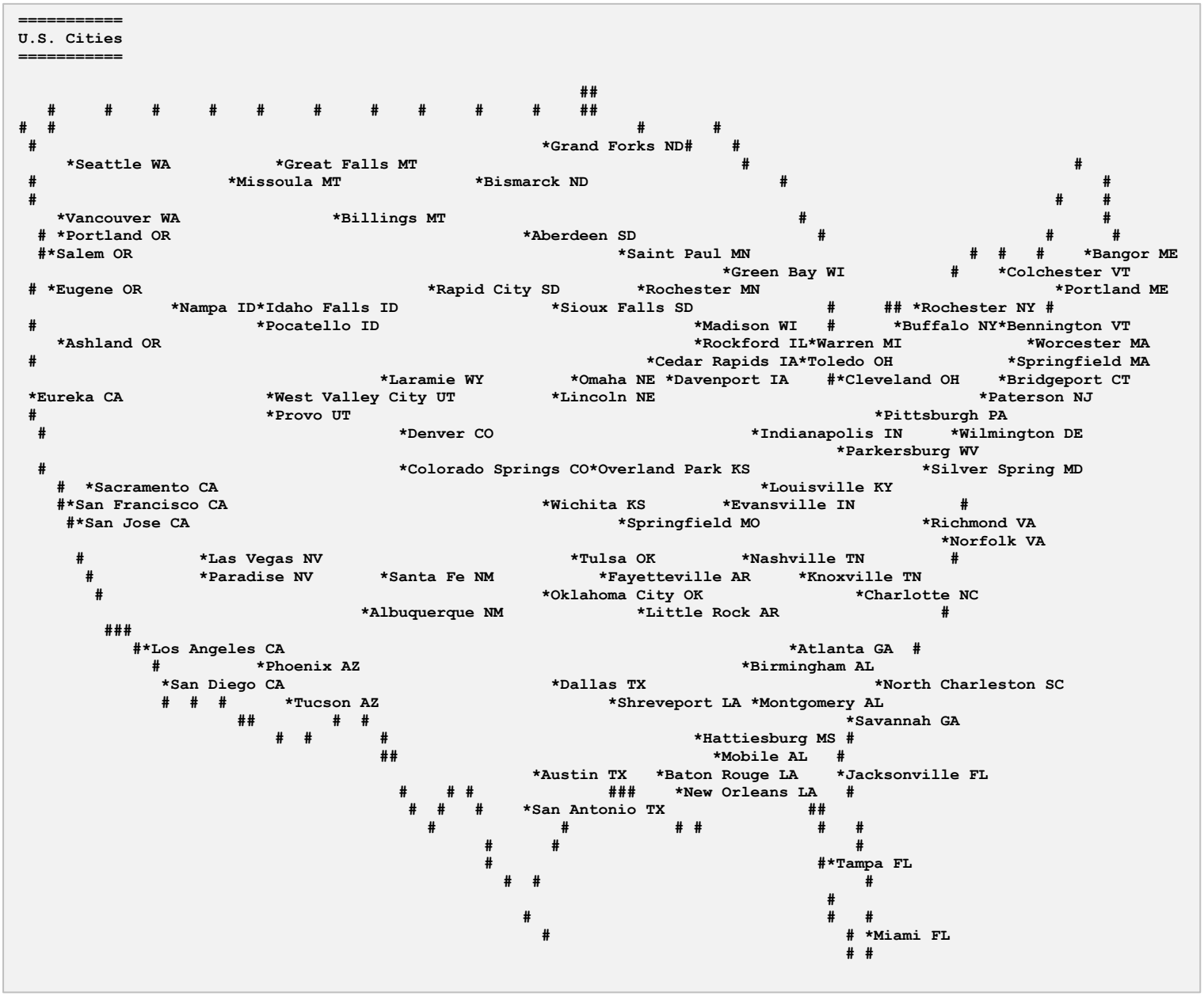


To generate the map, your program must read the file and create a **`Coordinate`** object from each input line. Class **`Coordinate`** must have a friend that overloads the input stream extraction operator **`>>`** to read each coordinate. Your program must create a **`Node`** object from each **`Coordinate`** object. The **`Node`** constructor must convert each coordinate to a print position. You are provided the **`Node`** member function **`convert_coordinate`** that maps a latitude to a row number and a longitude to a column number.

Your program must use a sorted linked list that is an instance of class **`SortedLinkedList`**. It must enter each **`Node`** object into the correct position of the linked list, which it must maintain sorted by row number and then by column number (similarly to the way you would sort a list of people's names by last name and then by first name). Once your program has read all the boundary coordinates, it should be able to iterate over the entire list and generate the boundary map. Class **`Node`** must have a friend that overloads the output stream insertion operator **`<<`** to print each **`Node`** object, and class **`SortedLinkedList`** must have a friend that overloads the **`<<`** operator to print the entire list. Class **`Node`** must also overload the **`>`** operator which **`SortedLinkedList::insert`** uses to compare two nodes when determining the correct insertion position for a new node.

U.S. cities map

Next, your program must input file **`city-data.csv`** to print this cities map:



To generate the map, your program must read the file and create a **`City`** object from each input line. Class **`City`** must overload the input stream extraction operator **`>>`** to read each city. Your program must create a **`Node`** object from each **`City`** object. Again, the **`Node`** constructor must convert each coordinate to a print position.

Enter each **`Node`** object created from a **`City`** object into the sorted linked list that you had created earlier with the boundary **`Coordinate`** objects. Continue to maintain the sort order by row and then by column. Your city **`Node`** objects will become interspersed with the boundary **`Node`** objects already in the list. Once your program has read all the city data, it should be able to iterate over the entire list and generate the cities map, which includes the boundary.

The overloaded **`<<`** operator of class **`Node`** must be able to output either a boundary node or a city node. Note that printing a city overrides printing the boundary. If multiple cities overlap in the same print row, printing the westernmost city has priority. Therefore, some cities may not be printed.

CodeCheck will match each map against master copies.

Destructors
Your program must clean up after itself before it terminates by destroying all the objects that it had created.

Submission into Canvas
When you're satisfied with your program in CodeCheck, click the "Download" link at the very bottom of the Report screen to download a signed zip file of your solution. Submit this zip file into Canvas. You can submit as many times as you want until the deadline, and the number of submissions will not affect your score. Only your last submission will be graded.

Submit the signed zip file from CodeCheck into Canvas: **Assignment #7. U.S. Maps**

Note: You must submit the signed zip file that you download from CodeCheck, or your submission will not be graded. Do not rename the zip file.

Rubric
Your program will be graded according to these criteria:

Criteria	Max points
Good output (as determined by CodeCheck) • Boundary map • Cities map	**35** • 15 • 20
Good program design • Class **`Coordinate`** • Class **`City`** • Overloaded **`Node::operator >`** member function • Overloaded **`operator <<`** friend function for class **`Node`** • Overloaded **`operator >>`** friend function for class **`Coordinate`** • Overloaded **`operator >>`** friend function for class **`City`** • **`SortedLinkedList::insert`** member function • **`Overloaded operator <<`** friend function for class **`SortedLinkedList`** • Object cleanup	**85** • 10 • 10 • 10 • 10 • 10 • 10 • 10 • 10 • 10 • 5
Good program style • Descriptive variable names and meaningful comments. • Follow the coding style (formatting, braces, indentation, function declarations before the main, etc.) of the Savitch textbook.	**10** • 5 • 5

Academic integrity

You may study together and discuss the assignments, but what you turn in must be your individual work. Assignment submissions will be checked for plagiarism using Moss (http://theory.stanford.edu/~aiken/moss/). **Copying another student's program or sharing your program is a violation of academic integrity.** Moss is not fooled by renaming variables, reformatting source code, or re-ordering functions.

Violators of academic integrity will suffer severe sanctions, including academic probation. Students who are on academic probation are not eligible for work as instructional assistants in the university or for internships at local companies.