CSC 415 Spring 2014

Assignment 3 – Introduction to C++

Part 1 Due: Tuesday February 11, 2014 by 11:59 p.m.
Part 2 Due: Friday February 28, 2014 by 11:59 p.m.
Grade: 45 points, penalty 10% per day late

Objective:
The objective of this assignment is for students to develop proficiency in programming constructs, functions, arrays and classes in C++, and become more familiar with Unix-based operating systems and IDEs.

Requirements:
You must design and develop the solution for this assignment **individually**. General questions about design and programming concepts, requirements or compile error messages may be asked publicly in class or on Piazza. Students are encouraged to respond to such questions without waiting for my intervention. Specific questions about your code should be addressed to me directly, either through 'private' posts on Piazza, after class, or during my office hours.

Assume that you are a software engineer at GF Software Solutions, Inc. and that your company has been hired to develop a system for Quirky Analytics, to perform various forms of analyses on large amounts of data. One task assigned to you is to develop a module to predict the likelihood of a site being contaminated, based on data available for nearby locations. For this assignment, you will develop a prototype system to help you and your customer better understand the problem and potential solution. The specifications for the prototype are discussed below.

Design a class called `Site` that stores data about a specific site. The attributes for the class should include at a minimum: Site Id, Site Name, Status, Remediation Status, Type of Contamination, X Coordinate, Y Coordinate.

Design another class called `SiteList` that uses a fixed size array to store all the sites in the system, i.e. objects of type `Site`. For the purposes of the prototype, assume that there will be at most 600 unique sites.

The prototype system should allow the user to:

- Input data about sites from a text file. Assume that the file has comma-separated values, i.e. each field in the file is separated by commas. Also assume that the data in the file has valid values and data types.
- Manually input data for a new site. A new site is added only if a site with the same name does not already exist in the system (i.e. there should be no duplicates).
- Input a set of coordinates (X and Y) for a location to see its status. If the site exists in the site list, the system returns the data in the "Status" attribute. Otherwise, the system makes a prediction based on available data and returns that value. See "Prediction Algorithm" below for further explanation.
- Write data from the list of sites, including new sites, to a text file in the same format as the input file.

File Structure:
The first line of the input file indicates the headings for each field and is for your information only. This line should be ignored when reading data into the array.

Each subsequent line represents one site, with each field (attribute value) separated with a comma. The last item on the line is delimited by the newline (`/n`) character or carriage return (`/r`) character.

The X and Y coordinates represent the geographic position of a site. For the prototype system, assume that the range for the X Coordinate is 375407 – 495843, and that for the Y Coordinate is 477499 – 572842.

Do not modify this file since I will use it for testing your implementation. You may manipulate the data as needed after reading it from the file.

Prediction Algorithm:
The potential status of a site that is not already in the system can be predicted by using a simplified version of the nearest neighbor classification algorithm.

Calculate the distances between the input site and sites in the system using the formula below:

$$distance = \sqrt{(x_{is} - x_i)^2 + (y_{is} - y_i)^2}$$

where

x_{is} and y_{is} are the coordinates of the input site and
x_i and y_i are the coordinates of the site i.

Then find five sites at the shortest distance from the input site.

Use the status of these five sites to predict the status of the input site. For example if all five sites are "Clear", then it is highly likely that the input site is also "Clear".

Part 1: Analysis due February 11
Using correct UML syntax, develop a class diagram for the structural design of this system. Consider how you can appropriately encapsulate data to facilitate information hiding and reuse. Your class diagram must show the relationships between the classes correctly. Be sure to label the diagram with your name and assignment number.

Use LucidChart or a similar UML-aware tool to draw the class diagram. Export the diagram as a `pdf` document. No other format will be accepted. You may sign up to join my LucidChart account at https://www.lucidchart.com/e/pulimood.tcnj.edu.

I will review the class diagram and give you feedback. Be sure to address this feedback in your final implementation. If required, submit a revised class diagram with Part 2.

Part 2: Implementation due February 28
Implement the classes and functionality described above, and write a simple driver to test your implementation. A text file, `sites.csv`, has been provided for testing purposes, but the user should be allowed to specify the names of the input and output files.

Handle errors gracefully. For example, the program should not crash if the user inputs data of an incorrect data type, or the number of sites exceeds the capacity of the array.

Document every source code file submitted:

- Include identification information at the top of every file;
- Document every class definition and method;
- Name identifiers well so that your code is "self-documenting"; and
- Provide adequate comments for complex code fragments or algorithms.

Create a script file that demonstrates that your program compiles without errors or warnings, and executes without runtime errors. The script should also demonstrate all the functionality and error-handling capabilities that have been provided.

General Instructions:

Before starting to work on this project, review Chapters 1 – 8 in the Zyante textbook, the slides, and class and homework exercises.

Follow the Guidelines for Programming Assignments posted on Canvas at:
tcnj.instructure.com/courses/1145055/wiki/guidelines-for-programming-assignments.

Your program must:

- Be implemented in C++ and follow best practices for software development,
- Include appropriate documentation, including for identification and maintenance,
- Implement a class, `SiteList`, that has a fixed size array as one of its attributes,
- Manipulate arrays extensively,
- Implement selection and iteration constructs,
- Read data from and write output to user-specified files,
- Enable user interaction (from the keyboard),
- Use appropriate formatting commands to display data neatly,
- Handle errors, such as user input and incorrect data types, and
- Be compiled on the Mac or Linux operating system using g++.

Deliverables:

Part 1:

The UML class diagram that models the structure of the system, submitted as a `.pdf` file, under 'Assignment 3 Part 1' by February 11, 11:59 p.m.

Part 2:

Zipped together and submitted as a `.zip` file under 'Assignment 3 Part 2' by February 28, 11:59 p.m.

- Script showing successful compilation and execution of your program.
- Complete source code for your program.
- A "readme" text file with a brief description of your program, and how to use it.

The revised UML class diagram, if required, resubmitted as a `.pdf` file, under 'Assignment 3 Part 1' by February 28, 11:59 p.m.

Grade:

As per the rubric provided.