CSCE 2014 – Programming Project 1

Midpoint Due Date – 08/25/2022 at 11:59pm
Final Due Date – 09/01/2022 at 11:59pm

1. Problem Statement:

Restaurant food delivery companies like DoorDash, GrubHub and UberEats have transformed how we buy and eat our meals. Instead of going to restaurants, we can now visit a website to order our meals and have food delivered. As we all know, some meals are better suited to home delivery, and others simply do not survive the trip from the restaurant to your dinner table.

The goal of this programming assignment is to create a data structure to store reviews of restaurants that offer home delivery. Specifically, your task is to design, implement and test two C++ classes. The "Review" class will store information about one customer's review of a restaurant meal delivered to their home. The "ReviewDB" class will contain a collection of Review objects. Your main program will have an interface that lets the user submit a review for a restaurant meal, search for all reviews for a given restaurant, or search for all reviews of restaurants that serve a given category of food (i.e. Pizza, Chinese, Burgers, Mexican). Your program requirements are described in detail below.

Task1: Create the Review class.

The data fields you must store in the Review class are: reviewer name, restaurant name, food category, delivery cost, and the customer ratings between [1..10] for delivery time, food quality, and overall satisfaction. Your Review class should have the normal constructors, getter and setter methods, and a print method. To test your Review class, you should create a test function that creates a Review object and makes a variety of calls to Review methods. You should call this test function from within your main program.

Task2: Create the "ReviewDB" class.

Once the Review class has been implemented, your next task is to create ReviewDB class. The private data in this object should be a fixed size array of Review objects and a count of how many reviews are currently stored in this array. To interact with this array of Reviews, you will need to implement the following four methods:

insertReview – adds a new review to the end of the array and increments the count.
printRestaurant – prints all customer reviews for a specified restaurant.
printCategory – prints all customer reviews for a specified food category.
printRecent – prints the N most recent customer reviews.

To test your ReviewDB class, you should create a test function that makes a number of "hard coded" calls to insertReview followed by a variety of calls to print methods. You should call this test function from within your main program.

Task3: Create the interactive program.

Your final task is to implement a main program with a text-based menu that allows users to insert reviews into the ReviewDB object and then print restaurant reviews using the three print methods described above. When your program starts, it should print the full command menu to the user (e.g. Enter '1' to insert review, Enter '2' to print reviews by restaurant, etc.). Your program should then loop reading and processing commands until the user chooses to quit the program. To test your program, you should enter a sequence of user commands that demonstrate all of the ReviewDB methods and save the output for your project report.

2. Design:

Your main design task before implementing the Review and ReviewDB class will be to decide on the names and data types for all of the private variables, and the names and parameters for all of the public methods. You may want to type the class definitions into "review.h" and "review_db.h" as you make these decisions.

Your second major task is to work out the algorithms you plan to use for the ReviewDB methods. Specifically, you will need to decide what loops to use, and what data comparisons are needed to select the correct reviews to print out.

Next, you need to design the text-based menu interface for your main program. In particular, you need to choose the letters or words the user will have to enter for each command, and what prompts will be needed to get user input.

Finally, you also need to decide what to do if the user enters an undefined command or invalid review information. For example, if a user enters a rating outside the [1..10] range you could ask the user to enter the data again, or you could "auto correct" their input in some way.

3. Implementation:

To implement your project, you should break your code down into five files: review.h, review.cpp, review_db.h, review_db.cpp, and main.cpp. If you use NetBeans, you can create the Review and ReviewDB classes using the method described in lab. If you create your program using OnlineGDB you will have to create these five files manually.

As always, it is a good idea to start with "skeleton methods" to get something to compile, and then add the desired code to each method incrementally writing comments, adding code, compiling, debugging, a little bit at a time. Once you have the methods implemented, you can work on the test functions and finally the main program that calls these methods to complete your project.

Remember to use good programming style when creating your program; clear names for variables and constants, proper indenting for loops and conditionals, clear comments, etc. Also, be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

4. Testing:

Test your program to check that it operates correctly for all of the requirements listed above. Also check for the error handling capabilities of the code. Try your program with several input values, and save your testing output in text files for inclusion in your project report.

You may want to create a text file that has a sequence of commands typed into it to simplify your program testing. You can “copy/paste” this file into your program’s input window to save yourself a lot of typing. The output may look a little strange because all of the program output will be printed after all of your input.

5. Documentation:

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Does it work properly for all test cases? Are there any known problems? Save this report to be submitted electronically.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted.

When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and all of your C++ program files. Do NOT upload an executable version of your program.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,
- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.