CS-217 Object Oriented Programming

Assignment 3

Topics: Classes, Operator Overloading

Release Date: 24 May
Due Date: 3 June 2021

Instructions

Submission: Combine all your work in one .zip file. Use proper naming convention for your submission file. Name the .zip file as **SECTION_ROLL-NUM_01.zip** (**e.g. A_20i0412_01.zip**). Your zip file should contain separate folders for each question, but no subfolders within each folder. Each folder should contain the .h and .cpp files for the question. Submit the .zip file on Google Classroom within the deadline. Failure to submit according to the above format would result in **25% marks deduction**. Submissions on the email will not be accepted.

Plagiarism: Plagiarism cases will be dealt with strictly. If found plagiarized, both the involved parties will be awarded zero marks in this assignment, all the remaining assignments, or even an **F grade** in the course. Copying from the internet is the easiest way to get caught!

Deadline: The deadline to submit the assignment is **3 June 2021 at 11:59 PM**. Late submission with marks deduction will be accepted according to the course policy shared earlier. Correct and timely submission of the assignment is the responsibility of every student; hence no relaxation will be given to anyone.

Bonus: In case you implement any additional feature which you think is worth of bonus, make it prominent so that we can see it at runtime.

Note:

- *Each question will be graded on the basis of your effort; additional marks will be awarded for using good programming practices, including: memory efficient programs, well-written, good design and properly commented.*
- *For all questions, it is important that you create each class in a separate .h file and include it in the .cpp file that contains your main function. The code should link correctly when run.*
- *For all classes, all data members should be private. All functions should be member functions except those used for initial data generation.*

Question 1: Recommender System

You are required to make a very simple recommender system for an online store. Your system should maintain a purchase history for each customer, i.e., that customer X bought item Y. When a customer logs into the store, they should see items related to the one they bought in the past. For representing relatedness between items, you can randomly generate an integer, representing a category, between 1-10 for each item. All items having the same category are related. For example, a pair of wired headphones, Bluetooth speaker, and a wireless headset would all have the same category and hence the same integer value associated with them. These would be related items and if a customer has bought one of them in the past, your system should recommend all the other items in the category when the customer logs into the store.

Your tasks are: (1) to write classes to implement the above recommender system, (2) generate some initial data for the system, and (3) write an interactive menu-driven program to demonstrate your system.

Classes:

You should create a number of classes. The bare minimum is a Customer class and an Item class. Each new customer and item in your system should be given a unique and increasing ID, i.e. the first customer gets ID 0, the second gets ID 1 and so on. Each customer should have at least a name and ID, and each item should have a name, ID and category. You may implement other classes and data members as required. All functions you write should be member functions. Global variables are not allowed.

Data generation:

At the beginning of your main function, **randomly** generate data for 25 customers, 30 items and a purchase history of at least 2 items per customer. There should be at least 10 item categories and 3 items within each category. Display this data neatly at the beginning of your program.

Menu-driven interface:

Next, implement a simple login function where a customer can enter his ID. After validating that the ID exists, you should display a menu that gives the following options:

1. View all available items
2. View purchase history
3. View recommended items
4. Login again (i.e. log out and log in as a different customer)
5. Exit

After a customer completes any action 1 – 4 from the above menu, display the menu again to allow the customer to perform any of the actions again, until the customer chooses to exit. In option 4, if a user logs in as a different customer, you should then maintain a separate purchase history for the new customer. Likewise, recommendations should be personalised separately for each customer. You are not required to simulate any new transactions; simply recommend items based on the past purchase history that you have randomly generated.

Question 2: Personal Health Record Management System

A personal health record (PHR) is a state-of-the-art paradigm in smart health care, where all of a patient's medical history is stored in a single, connected, online record. It is made up of different sections such as personal details (name, address, contact number etc.), medical history (e.g. regular medications), list of doctors that treat the patient, details of past medical visits, prescribed medications, etc. The objective of a PHR is to make all of a patient's history available in a single location, as opposed to being stored separately on paper-based files in different institutions. A doctor seeing the patient can access the patient's PHR to view all of the patient's medical history in one place.

You will create a simplified version of a PHR management system that allows very basic maintenance and access control of patient PHRs. A skeleton program has been created for you in the file phr.txt (attached). You may add more data members and functions to the given classes as required, or change the types of the existing ones. You will then separate all the classes into individual .h files and write a separate .cpp file containing main and any global functions.

When a patient needs to visit a doctor, he should add the doctor to his PHR. Any doctor should be able to request access to view any patient's PHR, but access should only be granted if the doctor is on the patient's list of doctors (in the patient's PHR); if not, the doctor should get the message "Access Denied". When the patient actually visits a doctor, a new medical visit instance should be created, with a unique ID, and all information (e.g. time of visit, doctor, institute, medicine) pertaining to the visit should be added to the patient's PHR (for details, see the medical_visit and PHR classes in the attached phr.txt file). A patient should also be able to remove a doctor from their PHR. If a doctor is removed, he should no longer be able to access the patient's PHR.

Set all values using setter funcitons and retrieve them through getter functions; all data members should be private. All functions, except for the initial functions to generate and display data, should be member functions. Global variables are not allowed, except for constant integers used to limit the size of arrays for storing the initial data generated for the system (see phr.txt file for details).

Your task in this question is to complete the driver program that is given in phr.txt. The full driver program is given, but you need to implement the functions that are called from it. You can add to the program but cannot remove anything from it. Read the comments carefully and implement everything that the comments require.

Question 3: Calendar Application

In this program, you will develop a calendar application that can display the full calendar for any month of any year and allow the user to add notes against any date. Implement the following functionality:

1. Display calendar for any month of any year
2. Add a note against any date
3. View all notes sorted by date
4. Edit or delete existing notes
5. Calculate the difference in months, weeks or days between any two dates
6. Calculate the date and day of the week n months, n weeks, or n days from a given date.
7. Exit

When the program is run, the calendar of the current month (taken from the system) should be displayed by default. Then the user should be given the following options:

1. **Display calendar for a different month**
   Take user input for the month and year whose calendar he would like to view and display it.

2. **Manage notes (View, add, edit or delete)**
   Ask the user whether to view, add, edit or delete notes. View notes should allow viewing all notes or retrieving notes against a specific date. For adding, editing, or deleting, the user will need to enter a date on which to add, edit or delete a note.

3. **Calculate difference between two dates**
   Take user input for both dates and calculate the difference between them in months and days as well as weeks and days. For example, if the user enters x and y, display the difference as:

   39 months, 3 days, OR 78 weeks, 3 days.

4. **Calculate a future date**
   Take user input for the start date and for the number of weeks, months or days to add to it. The user can type “3 months”, “43 weeks”, “237 days” etc. and your program should interpret the text input correctly and add the appropriate number of days to the start date. Display the output in the form:
   <user input> from the start date <user input> is <day of the week>, <day> <month> <year>.
   For example:
   43 weeks from the start date 3 March 2021 is Thursday, 6 January 2022.

5. **Calculate a past date**
   This is the reverse of part 4 above. Take user input for the start date and subtract the number of days, weeks or months that the user specifies.

Implement the above as member functions in at least a **Date** class and a **Calendar** class. You can use any number and type of data members and also implement additional classes if needed.

Write a driver program that interacts with the user and gives options to test all the above functions; the program should keep displaying the options until the user chooses to exit.

Question 4: ArrayInt Class

Implement an ArrayInt class that allows you to easily create and manipulate integer arrays. Implement the following functions in the class:

- ArrayInt(int); //parameterised constructor; it should create an integer array of the given size.
- Copy(ArrayInt); //the array passed as argument should be copied into the calling array
- makeNull(); //the calling object should be cleared, i.e. all locations made empty
- initialise(int); //all values in the calling array should be initialised to the argument passed.
- sum(); //returns the sum of all values of the array
- average(); //returns the average of all values of the array
- contains(int); //checks if the array contains the integer passed as argument
- getIndexOf(int); //returns the index of the integer passed as argument; if it does not exist in the array, returns -1
- getElement(int); //returns the element at the index int of the calling array.
- getSubArray(int, int); //returns a sub-array of the calling array, i.e. the elements between the index given in the first and second argument. For example, getSubArray(5, 9) should return an array of the elements between location 5 and 9 of the calling array.
- sort(); //sorts the values in ascending order
- reverse(); //reverses the array values

Additionally, overload the following operators for this class:

operator+

This should create a new array in which the two operands are concatenated.

operator*

This should element-wise multiply the two arrays; i.e. element i of array 1 should get multiplied by element i of array 2. The result should be stored in a third array.

operator==

This should check if two arrays are exactly the same, i.e. all elements are same.

operator>

This should check if the sum of all values of the calling array is greater than the sum of all values of the argument array.

Write a driver program that creates some ArrayInt objects and clearly demonstrates all of the above functions.

Question 5: Matrix class

In this program you will create a Matrix class for representing 2D matrices of different sizes and demonstrate the use of some overloaded operators for easy manipulation of the matrices.

Create a Matrix class with a dynamic 2D array as a data member. The paramterized constructor should allow creating a matrix (i.e. a 2D array) of any size (e.g. 6X4, 2X3, 5X5, etc.). Add additional functions in your class such as getters and setters as needed, as well as a display function. Data members must be private. You must write a destructor.

You will overload the following operators (as member functions) for operations on objects of the Matrix class:

Operator+

Adds two matrices together. Returns the resulting matrix or exits with the message "Invalid operation" if the matrices are different sizes.

Operator-

Performs matrix subtraction; returns resulting matrix or prints an "invalid operation" message.

Operator*

You will write two versions of this operator:

1. Scalar-matrix multiplication: multiplies a matrix with a scalar, i.e. a single real number, and returns the resulting matrix.
2. Matrix-matrix multiplication: multiplies two matrices together after checking that their sizes allow multiplication; returns the resulting matrix of prints an "invalid operation" message if their sizes are not compatible.

Operator==

Compare two matrices element-wise; return true if all elements are equal, and false otherwise. Also return false if the matrices are different sizes.

Operator++

You will write both prefix and postfix versions of the increment operator. Both will increment each element of the matrix, but make sure they work in a way that is consistent with the usual working of the prefix and postfix increment, i.e. the prefix increment should increment the matrix first before executing the rest of the statement, and the postfix increment should first execute the statement and then increment the matrix.

Operator=

Copy one matrix into another. Be mindful that by default the = operator performs a shallow copy; in case of dynamic data members, you need to write functionality for "deep" copy, i.e. both objects should occupy separate areas in the heap.

Write a driver program that takes user input for the sizes and elements of 2 matrices. Then call all of the above operators for the two matrices and display the results. The file

sample_matrix.pdf shows a sample output for a run of the program; please refer to it when writing the driver program.