Project 2 – Programming Project

CS3330 Data Structures and Algorithms
Term 5 2018: May 29 – July 29
Dr. Jack Davault

Overview

This assignment consists of implementing an application using the techniques learned in the first half of the course. Examples on how to use file operations, the standard template library, and other features are provided in the *Sample Programs (for Project 2)* section located the *Learning Modules* area in Blackboard. These examples provide a starting point for your implementation. Everything you need to do this project is provided in these sample programs.

Your program must compile and you must provide all of the source code files so that I can also compile and run your program (i.e. provide me with all files ending in **.h** and **.cpp** that are necessary to compile and run your program). You are responsible for submitting your program correctly. Please ask if you have questions.

You may use any resources, books, or notes. **Apart from your textbook and provided examples, you must mention all resources that you use as source code comments next to the applicable lines of code or the appropriate functions within your source code or within the main comment at the top of your source code file.** You may also work with only one other classmate on this assignment if you want to, but you must mention all of your names as comments in the applicable sections of the source code showing where you collaborated. If you collaborate on the entire project, no problem, both of your names must be in the comment at the top of all of the source code files, and only one person needs to submit the assignment. **I do highly recommend that you work with one other person on this project, and begin working on this assignment as soon as possible. Don't wait until the last minute to begin working on this assignment.** Consider posting a message in the *Student Lounge* or sending out an email to your fellow classmates asking if anyone is willing to partner with you on this project. Two person teams only.

Note: Again, you may use outside resources or books, but you are not to post questions, comments, or source code related to this assignment in any public open forum other than the ones specified for this course in Canvas (e.g. *Ask the Instructor* or *Student Lounge*). If you like, you can also set up a Group Set in Canvas. Note that pay for websites for programming solutions are not an acceptable method for completing this assignment. If you have a question, don't sit on it for too long; ask me in the *Ask the Instructor* forum or via e-mail. I am also available by appointment via *Skype*.

This assignment weighs 15% of your final course grade. When submitting your work, be sure to Zip up all source code files and documentation (if any) into a single file.

Implementation (25 points)

Implement a simple hotel reservations system as a proof-of-concept prototype for fictitious project to be used on a larger system. The system you are developing must provide a simple text based interface that will allow the user to enter attributes associated with each reservation. The program will manage the attributes for each reservation, which will be an individual node within a single linked list stored in memory. The linked list can be a global variable for simplicity like the Book Inventory example. The program must use the "list" API in the C++ standard template library (STL). The program must implement at least one class that will hold the following variables (input validation is only required where indicated):

Field Name	Data Type	Description
• `locaterId` –	string	An automatically generated string to identify the locator identification number for the reservation. The string must be in the form **LLNNN-LL**, where **N** is a number from 0 to 9, and **L** is a character from A to Z. For example: MQ406-GF. The `locaterId` must a unique identifier for each reservation in the linked list and repeats are not allowed.
• `checkInDate` –	string	The date of that the guest will check in to the hotel. The date must be single string in the form **mm/dd/yyyy**. Where **mm** is the two-digit month, **dd** is the two-digit day, and **yyyy** is a four-digit year. ***Input Validation Required:*** The program must re-prompt the user to re-enter the date if it is not in valid format provided above.
• `checkOutDate` –	string	The date of that the guest will check out of the hotel. The date must be in the form **mm/dd/yyyy**. Where **mm** is the two-digit month, **dd** is the two-digit day, and **yyyy** is a four-digit year. ***Input Validation Required:*** The program must re-prompt the user to re-enter the date if it is not in valid format provided above.
• `numberOfDays`	int	The total number of days that guest will be staying.
• `numberOfOccupants` –	int	The number of occupants who will be staying in the room.
• `rate` –	double	The dollar amount charge per day for the room.
• `roomType` –	int	The value of this field must range from 1 to 6 to associates the room with one of the following valid types: 1=One king size bed, 2=One king size beds, 3=One queen size bed, 4=Two queen size beds, 5=Suite, 6=Special Request ***Input Validation Required:*** The program must re-prompt the user if the value of the entered type is not within the valid range.
• `firstName` -	string	A string variable to hold guest's first name.
• `lastName` -	string	A string variable to hold guest's last name.
• `street` –	string	A string variable to hold the guest's street name and street number. Spaces must be allowed between the street number and street name.
• `city` –	string	A string variable to hold the name of the guest's city.
• `state` –	string	A string variable to hold the two-character state abbreviation of where the guest resides. ***Input Validation Required:*** The program must re-prompt the user if the entered state if it is not two characters.
• `zipCode` –	int	Holds the zip code for where the guest resides. ***Input Validation Required:*** The program must re-prompt the user to re-enter the zip code if it is not a five-digit number.

• `emailAddress` –	string	The guest's e-mail address. ***Input Validation Required:*** The program must re-prompt the user for a valid e-mail address if the entered string does not contain an '@' character.
• `phoneNumber` –	string	A string variable to hold the guest's contact phone number.

Provide the appropriate methods to set and get the data for each of these class variables. For example `setFirstName(string firstName)` and `string getFirstName()`. The Book Inventory provides an example on how this can be done. In addition, the main program must provide the following functionality:

1. When the program is first started, it must read a data file called **bookings.dat**. The program must not prompt the user for the name of the data file. The name of the file must be hard-coded within the program. If the file exists, the program will load the data for each reservation into a global linked list. If the file does not exist, the program will start with an empty linked list.

2. The program will provide a simple text-based user interface that manages the all of the reservations within a linked list. Each reservation must be placed in the linked list as an object that holds all of the attributes associated with it as mentioned above. The user interface will allow the user to perform the following:

   (a) Enter Reservation – allows the user to enter all of the fields for new reservation, except for the `locaterId`, which will be automatically generated by the program as previously mentioned. After the fields are entered, the program will place the reservation object in the global linked list.

   (b) Display Reservations – displays all of the reservations within the linked list along with their associated properties. When displaying the room type, the program must print the room type number along with the associated description (e.g. "One king size bed", "Two king size beds," etc.). In addition, this option must print the total charge for each reservation based on the rate and number of days that the guest will be staying. Also, print the total number reservations in the linked list.

   (c) Find Reservation – allows the user to find a reservation by its `locaterId`. The program will prompt the user to enter the `locaterId` and will display all of the fields associated with the given reservation if it is found. The program must display an error message if the reservation is not found in the linked list.

   (d) Modify Reservation – allows the user to edit the fields for a given reservation that is in the linked list (except for the `locaterId`). The program must prompt the user to enter the `locaterId` as the key to find the reservation to edit. Print a message if the reservation is not found in the linked list. For simplicity, the program may re-prompt the user to re-enter all of the fields associated with the given reservation; however, it must reuse the `locaterId` value.

   (e) Cancel Reservation – allows the user to delete a reservation from the linked list using the `locaterId` as the key. The program must display a message if the provided `locaterId` is not found in the linked list.

   (f) Exit System – before the program exits, it must save all of the reservations in the linked list to the data file. I recommend using a standard text file with one field in the object per line. At this point, if the file does not exist, the program will create it.

Hints and Tips

The Book Inventory program provides a good place to start, and I recommend using it as a model and starting point for your program. It provides a simple text based interface, shows how to create a class, and how to use the C++ STL library's linked list API for adding and displaying items in a linked list. You should rename its files, the class, and methods as a starting point for your program.

Start by breaking the program down into small pieces. First, work on the feature that allows you to enter a reservation along with all of its fields. Next, work on the display feature that will show all of the reservations and their associated fields contained within the linked list. Next, make sure your program can read and write one reservations and its fields to and from the data file. Then expand the program to read and write all of the reservations to and from the data file using appropriate loops. The data file functions for this can be modeled after those in File Operations example and the display and enter reservation functions that you will create. Note that the date File Operations example uses an array. Your program will use a linked list. The loadData() function can be modeled after the display function, except you will write to the data file (e.g. the dataFile pointer) instead of the screen using cout. The saveData() function will be modeled after the enter reservation function, except you will read from the data file instead of cin. Finally, work on the search, modify, and delete features. Be sure to review the sample code in the *Sample Programs (for Project 2)* section in the *Learning Modules* area in Blackboard.

There are no tricks or special techniques required for this assignment. Everything you need to successfully create this program is available in the provided sample code.

Other Comments

Please do not hesitate to let me know if you have questions in the *Ask the Instructor* forum or via e-mail. I also encourage you to discuss this project among yourselves in the *Student Lounge* area. Again, you may not post questions in any public open forums for this project. It is not necessary, and typically their solutions are wrong. You must work within this bounds of the forums within this class: The *Ask the Instructor* and *Student Lounge* or work with one other student on this project, if possible.