Due Date and Submission Instructions

This program is due Friday, November 16th, at 11:59:59 pm. Submit your program files as attachments to an email to

cs2projsn@cs.rpi.edu

where n is YOUR section number. Use the EXACT subject line

project 2

Grading Criteria

This project is worth 120 points toward your homework grade. The points are divided up into 30 points for compilation, 40 points for structure and 50 for correct execution.

In your program structure be careful to NOT introduce a lot of interactions between your classes. (i.e. do NOT pass the list of videos to the list of customers (see below) or vice-versa.) Strive for a clean class design.

In the correct execution grade you will lose points for not following instructions. One example is improper use of command-line arguments.

Initial Hint

Read page 212 bottom through 218 in the Carrano book first. This assignment will be somewhat similar, and you might find it helpful as an example of how to go about designing an object-oriented solution to a problem. However, notice that you have to implement the program described in this assignment, NOT the one described in the book. Reading pages 18-24 in C&P or sections 4.10 and 5.13 in the Deitel and Deitel book might also be helpful for understanding how to design an object-oriented solution to a problem.

General Description

In this programming assignment, you’ll write a program for running a (simple) video store. Here’s a description of how the store works:
When a customer comes to the check out counter, the assistant asks if the customer has been there before. If the customer has not been to the store before, the new customer has to be added to the system: name and credit card number of the customer are required for this. If the customer is not new, the assistant asks for his name and retrieves the account for this customer.

After the assistant has entered the customer’s name into the system, the customer mentions one video title that he would like to check out. The assistant looks up if there are still copies of that video title available. The store has multiple copies of each video which are distinguished by the title of video together with the copy number. If there are copies left, the assistant gives that video to the customer and updates the inventory system accordingly (i.e. adds the information that this copy of this title has been checked out by this customer and the due date). If the store is out of copies of that video, then nothing needs to be done. Otherwise, the assistant tells the customer the total amount he has to pay. The customer is charged $3.99 for a 3-day rental of a new release, and $2.99 for a 5-day rental of a classic video. The customer also has to pay any outstanding balance of his account (late fees of previous rentals), if applicable. Assume that all customers pay the whole amount right away, so after check-out the customer’s account balance should be 0.00.

When customers bring back video tapes, they simply throw them into the drop box. When the shop assistants take the video tapes out of the drop box, the videos are identified by the video title and the number of the tape copy, and the inventory needs to be updated with the information that that particular tape was brought back. If that tape was overdue, the customer’s account should be charged with $2 per late day. The customer who rented this movie can be determined because that information was stored in the record for that video tape at check-out time.

**Data structure requirements**

1. You need a sorted list of video title records, and a sorted list of customer records. The sorted-list code should be implemented as a linked list in both cases. You may reuse and/or modify code from previous labs, homeworks, or code examples. Notice however that you’ll need
one or more functions for retrieving certain items (a particular video
title, a particular customer). Consider using a template class to im-
plement the sorted list, but you don’t have to implement it that way
if you find it too difficult. The video title list should be ordered alphabetically by video title (e.g. “casablanca” before “terminator”), the
customer list should be ordered alphabetically by name.

2. You’ll need a class to store the information pertaining to a video title,
and another to store the information pertaining to a particular copy of
a video (a tape). As explained in the description of the store, there are
several copies of each video title. The video title class should contain
a list of the copies of this video title (who rented this copy, when is
it due, etc). The list of copies (tapes) stored in the video title class
could be implemented as a linked list but the implementation will be
much easier if you implement it with a dynamically allocated array.

To recap: you have to implement the list of video titles with a linked
list, but you should implement the list of copies of each title with an
array. Assume that the name of the video title (e.g. casablanca,
cats & dogs) is a single word that does not contain spaces. Notice
that you’ll have to update a customer’s record if a video tape (a par-
ticular copy) is brought back late. Carefully consider how to store the
customer information in the video tape class so that doing this update
is fairly easy.

3. You’ll need another class to store the information pertaining to a cus-
tomer. To simplify this assignment, the customer record should NOT
contain a list of the movies that the customer has currently checked
out, only the necessary financial information (and the customer’s name
of course). Assume that the customer has only one name (not first and
last name), to simplify the assignment.

Determine from the store description above what data members you need
for these classes and determine if additional classes are necessary. Read the
description very carefully — all data pieces mentioned should be represented
in your program (due dates, customer balance, etc.). Decide deliberately
where to store which piece of information. It is important that you spend
some time in the design stage thinking about which information you need to
perform which of the functions described in the “user interface” section, and
which member functions with which interfaces your classes should implement
to achieve this. We strongly suggest that you draw a picture of all the data
structures involved — this will prevent you from getting confused, and will help you localize information. You may submit your design to the TA for critique if you want (but you are not required to do this — this time you only have to hand in the implementation since we won’t have time to grade a lot of designs), or simply ask him/her for advice if you have difficulties with the design.

Further details:

1. For this assignment you should not use any STL classes, but using C++ string class is a MUST.

2. To simplify matters, assume that there is a global integer variable today that contains the day of the year to represent the date (i.e. January 1 is day 1, and Dec 31 is day 365). At the start of the program, this day should be initialized with 100.

3. At the start of the program, the program should read the list of video titles from file (provided on the web page): the file contains information about each video title in the following form (one title per line):

   videotitle numberofcopies newrelease

   videotitle is a single word (string), numberofcopies is an integer, and newrelease is either 1 or 0. The name of the input file should be passed into the program as command-line argument. (A number of students did NOT do this for the last programming project and they lost points for it!) Thus, the program is run from the command line like this (assuming that the file with the video titles is in the same directory as the program):

   videostore.exe videotitles.txt

4. To further simplify matters, assume that at the start of the program, all videos are in the store, and the store does not have any customers yet.

5. Assume that the assistant enters all video titles and customer names in smallcaps.

6. You do not have to implement elaborate error checking, only what’s specifically indicated below. Also, add a few asserts whenever you feel it will help you debug your program.
User interface

The main function should offer the user (i.e. the store assistants) the following options in a loop that returns to this menu after finishing the respective task (with exception of the last one):

1. add new customer
2. check-out
3. return video
4. advance day
5. exit program

The user should type in the number (1-5) to select an option.
For the first three options, the program has to ask for additional information as necessary (see description of the store). For option 1, the program should ask for and input the name of the customer. Don’t add this customer if the name is already in the list. Thus, the program output and user input should look like:

Enter new customer name and credit card number:
john_smith 123456789
john_smith added as a customer

or

Enter new customer name and credit card number:
john_smith 123456789
Error: john_smith is already a customer

For option 2, assume that only a single video is to be checked out each time. If the customer wants more than one tape, the assistant will have to choose that option again, and the customer will get one bill for each tape. That’s not particularly realistic but it’ll keep the user-interaction code simple. Thus, in option 2 the program should ask for the customer name and the video name:

Enter customer name and video name
john_smith star_wars
This rental should be rejected if john_smith is not a customer, if star_wars is not a movie or if all copies of star_wars are checked out.

For option 3, the program should ask for the movie name and copy number.

Enter movie name and copy number
star_wars 2

Reject this if the movie name is wrong, the copy number does not exist, or the copy has not been checked out. Notice that the customer name is NOT available, only the tape title and copy number. Make sure that you update all data structures properly: the tape should be available after it’s returned, and the customer needs to be charged money if the tape was late.

For option 4, the program should simply increase the day variable. For option 5, the program should deallocate all dynamically allocated memory and then exit.

Final Notes

We will provide test files with user input for you so that you can test if your program fulfils the specifications. We will use very similar test files for the evaluation of your program. To feed input from a file instead of actually “interacting” with the program, call the program from the command line like this.

videostore.exe videotitles.txt < testinput.txt

where in this example, testinput.txt contains all the user’s input (but nothing else). We will use input from a file like this in testing your program. Example files for both the video title file and the user input file will be provided on the web page.