CSCI-1200 Data Structures — Spring 2014
Homework 4 — DVD Preference Lists

In this assignment you will write a program to manage the physical DVD media inventory of Netflix and ship DVDs to customers according to their specified movie preferences. Your program will handle several different operations: adding a number of copies of a DVD to the inventory, signing up a new customer with their initial preferences, shipping DVDs to the customers, restocking DVDs as they are returned by customers, modifying the customer preference lists, and printing information about specific DVDs and customers. We provide the code to parse an input file that specifies these operations. Furthermore, we provide the core implementation of the DVD shipping algorithm. You will use the STL `list` class heavily in your implementation of this program. *Please carefully read the entire assignment before beginning your implementation.*

The input for the program will come from a file and the output will also go to a file. These file names are specified by command-line arguments. Here's an example of how your program will be called:

```
preferences.exe requests.txt results.txt
```

The form of the input is relatively straightforward. Each request begins with a keyword token. There are eight different requests, described below. You may assume that the input file strictly follows this format (i.e., you don't need to worry about error checking the formatting). Here is a sample input file and the corresponding output file:

requests_small.txt

```
dvd "Into the Wild" 3
dvd "Forrest Gump" 2
dvd "Gone with the Wind" 2
dvd "Raiders of the Lost Ark" 1
dvd "Toy Story" 3
customer Carol Adams 4
  "Raiders of the Lost Ark"
  "Gone with the Wind"
  "Forrest Gump"
  "Into the Wild"
customer Kim Smith 3
  "Raiders of the Lost Ark"
  "Into the Wild"
  "Gone with the Wind"
customer Wayne Evans 1
  "Raiders of the Lost Ark"
ship
return_newest Carol Adams
return_oldest Kim Smith
add_preference Carol Adams "Toy Story"
ship
return_newest Carol Adams
ship
print_customer Carol Adams
print_customer Wayne Evans
print_dvd "Toy Story"
print_dvd "Gone with the Wind"
```

results_small.txt

```
3 copies of "Into the Wild" added
2 copies of "Forrest Gump" added
2 copies of "Gone with the Wind" added
1 copy of "Raiders of the Lost Ark" added
3 copies of "Toy Story" added
new customer: Carol Adams
new customer: Kim Smith
new customer: Wayne Evans
Ship DVDs
  Carol Adams receives "Raiders of the Lost Ark"
  Kim Smith receives "Into the Wild"
  Carol Adams receives "Gone with the Wind"
  Kim Smith receives "Gone with the Wind"
  Carol Adams receives "Forrest Gump"
Carol Adams returns "Forrest Gump"
Kim Smith returns "Into the Wild"
Ship DVDs
  Carol Adams receives "Into the Wild"
Carol Adams returns "Into the Wild"
Ship DVDs
  Carol Adams receives "Toy Story"
Carol Adams has 3 movies:
    "Raiders of the Lost Ark"
    "Gone with the Wind"
    "Toy Story"
Wayne Evans has no movies
  preference list:
    "Raiders of the Lost Ark"
"Toy Story":
  1 copy checked out and 2 copies available
"Gone with the Wind":
  2 copies checked out
```

And here are the details on each command:

dvd ***"movie title" copies*** Adds the specified number of copies of this movie DVD to the inventory. Movie titles may consist of one or more words, so they are always placed in double quotes. If the movie is already in the inventory, simply add more copies to the record.

customer ***first_name last_name num_movies "1st movie" "2nd movie" ... "nth movie"*** Each customer is described by two strings, their first and last names. The name is followed by an integer, the number of movies in their initial ranking, and then the titles of those movies in preference order. Your program should print a warning message if a customer of this name is already stored in the system or if the movie preference list contains duplicates, or includes movies not in the DVD inventory.

add_preference ***first_name last_name "movie title"*** This command allows the customer to add an additional movie to their preference list. The new request is added at the end of their preference list. A customer is allowed to request a movie they have already been shipped and returned. Your program should print a warning message if the customer already has this movie in their preference list, if no customer with this name exists in the system, or the movie is not in the DVD inventory.

ship This command simulates the shipment algorithm that is run once a day. The customers are organized by priority to receive a new movie. Higher priority goes to customers who have been waiting longer since they last received a new movie. The customers are examined in priority order. For each customer, if they do not already hold three movies (Netflix's original cap on the number of movies a customer could hold at one time) *and* they have at least one movie in their preference list, the algorithm attempts to ship them a movie. The algorithm looks through their preference list in order, from first choice down to last choice and ships the first movie that has at least one copy available (not held by another customer). If the customer is shipped a movie, the customer priority queue is rearranged to put this customer at the back. If none of their requested movies are available, the customer stays where they are in the queue. The algorithm repeats until each customer either has three movies or no copies of their requested movies are available.

return_oldest ***first_name last_name***
return_newest ***first_name last_name*** These two commands examine the movies currently held by the specified customer and return either the DVD they have held the longest (oldest), or the DVD they received most recently (newest). When a DVD is returned, that copy is again available to be shipped out to another customer. Your program should print a warning message if no customer with this name exists in the system, or if the specified customer does not currently have any DVDs.

print_customer ***first_name last_name*** Prints the DVDs currently held by the specified customer (in order from oldest to newest) and prints the DVDs remaining in the customer's preference list (from first choice through last). A warning message is printed if no customer by this name exists. *For extra credit:* When the "`--analysis`" command line argument is specified, customer satisfaction information for this customer is printed.

print_dvd ***"movie title"*** For the specified movie, prints the number of available copies and the number of copies currently held by customers. A warning message is printed if no movie with this name is in the inventory. *For extra credit:* When the optional "`--analysis`" command line argument is specified, rental information about this movie is printed.

All of the normal output of the program, as well as warning messages about invalid requests, are saved to the output file specified on the command line. *Note: Warning messages do not cause the program to crash or exit.*

Extra Credit

For extra credit, analyze the performance of the algorithm in satisfying customer requests. For example: How many movies were shipped to each customer? What percentage of DVD shipments were the customer's first choice? How many days was each customer waiting while holding fewer than 3 DVDs, with at least one DVD on their list, but no DVD was shipped to them? How many times was each movie title shipped? How many people request each movie title? Use the optional argument "`--analysis`" to specify that this additional information should be printed with each customer and DVD. Your code should also summarize the overall performance of the algorithm after processing the last request in the input file.

Discuss in your README.txt an alternate algorithm that you believe would better satisfy the customer demand. Implement your new algorithm and run the same customer satisfaction analysis. Paste the summary output information into your README.txt file and discuss your conclusions. Your new shipping algorithm should run when the optional command line argument "`--improved`" is specified:

```
preferences.exe requests.txt results.txt --analysis --improved
```

Assignment Requirements, Hints, and Suggestions

- **You may not use vectors or arrays for this assignment.** Use STL lists instead. You may not use maps, or sets, or things we haven't seen in class yet.
- We have provided a partial implementation of this program, focusing on input parsing and the DVD shipping algorithm. There are member function calls to our versions of the `DVD` and `Customer` classes, so you can deduce how some of the member functions in our solution work. You are *strongly encouraged* to examine this code carefully and follow the interfaces suggested in the provided code.
- In your README.txt file, provide an order notation analysis of each operation assuming d different movie DVD titles, an average or maximum of k copies of each DVD, c customers, and an average or maximum of p movies ranked on each customer's preference list.
- Do all of your work in a new folder named `hw4` inside of your homeworks directory. Use good coding style when you design and implement your program. Be sure to make up new test cases to fully test your program and don't forget to comment your code! Use the template `README.txt` to list your collaborators, your time spent on the assignment, and any notes you want the grader to read. When you are finished please zip up your `hw4` folder exactly as instructed for the previous assignments and submit it through the course webpage.