ECS 40 **Program #4 (50 pts, 2.5 hours)** **Spring 2016**

Due: Wednesday, April 27th, 11:59 PM Executable name: airline..out Handin to cs40a p4

Filenames: authors.csv, main.cpp, utilities.cpp, utilities.h, plane.cpp, plane.h, flight.cpp, flight.h, passenger.h, passenger.cpp, flights.h, flights.cpp, and Makefile.
New concepts: comma separated value files, binary files, vector, maintaining a sorted list, static consts, static functions, friend.

In this program you will have two new classes, and replace the reservations files with two pairs of files. You can find passengers.dat, flights.csv, my airline.out, and PrintDatFile.out in ~ssdavis/40/p4. You may use my code from p3 as your starting point if you wish. It will be in ~ssdavis/40/p3/SeansSrc on Thursday morning.

Specifications

1. Flights class
   1.1. The Flights class will contain a dynamically allocated array of Flight* class objects named flights, an int named capacity that stores how large the array is, and an int named size that keeps track of how many flights there are currently stored in the array.
   1.2. The constructor shall create the array with a capacity of 2 Flight*.
   1.3. A function named insert(Flight*), should be used to insert a Flight object pointer into its proper position in the array that is sorted by the flightNum of each Flight. If insert() tries to insert into a full array, then the array should be resized to twice its current capacity.
   1.4. addPassenger(), and readFlights() in main.cpp should now be methods of the Flights class.
   1.5. Part of writeFlights() and all of freeFlights() will be combined to form the Flights destructor. The destructor will write the number of flights to flights2.csv, then delete each Flight*, and then delete the whole array of Flight*. The destructor of Flight will append information about the given Flight to flights2.csv.
   1.6. There is now a menu item to add a flight specified by the user. All entries by the user are guaranteed to be correct, so there is no need for range checking of any kind. Each class should ask for, and store its own information. You will have to add a default constructor to Plane() for this to work.
   1.7. There is now a menu item to remove a flight specified by the user. Though the size will be reduced, the array is never shrunk in capacity. All entries by the user are guaranteed to be correct, so there is no need for range checking of any kind.
2. Passenger class.
   2.1. The Passenger class has the following data members: short flightNum, short row, char seat, and char name[30]. They must be declared in that order for the passengers.dat file to be read correctly
   2.2. Each Passenger will be stored in a binary file, passengers.dat, described below.
   2.3. Each element of the Plane::passengers array will now contain an int instead of a char*. The int will be the position of the passenger in passengers.dat. Use -1 to indicate that there is no passenger in a given seat.
   2.4. There may not be more than one Passenger object existing at any given time in the program! You will rely on searching the passengers.dat for information about a given Passenger.
   2.5. The Plane::addPassenger must add the new passenger to the end of the passengers.dat file. This will preserve the validity of the indices in the Plane::passengers arrays.
   2.6. There is now a menu item to remove a passenger. All entries by the user are guaranteed to be correct, so there is no need for range checking of any kind.
   2.7. When a passenger is removed through removal of the passenger or their flight, its flight number in passengers2.dat should be changed to -1. When writing passengers3.dat, removed passengers should not be written to the file.
3. I have changed the data files for the program. There is no longer a reservations.txt, nor reservations2.txt. There is now a flights.csv file, a flights2.csv, a passengers.dat file, a passengers2.dat file, and a passengers3.dat.
   3.1. flights.csv has, as its standard suffix indicates, comma separated values. Much the same as with reservations.txt, the first line contains the number of flights. Each subsequent line of the file contains all of the information for a flight in the following order: 1) flight number; 2) origin; 3) destination; 4) rows; and 5) width. Instead of using strtok() to dissect the later lines of the file, you should use getline().
   3.2. passengers.dat is a binary file of Passenger objects. Note that this binary file can only be accurately read by an Intel-like CPU based computer. All of the Passenger's that have reserved seats are in passengers.dat, but they are in

random order. When using sftp to transfer passengers.dat to your home, make sure that it is transferred using binary mode. All interactions with passengers.dat and passengers2.dat should be done from Plane methods.

3.3. passengers2.dat is a binary file that is initially a perfect copy of passengers.dat. You will create, and update this file as your program runs.

3.4. passengers3.dat is a binary file of Passenger objects, with the Passenger's sorted in the same way they were for programs #1 and #3—by flight number, row, and lastly seat. You will write this file when your program terminates. You can use PrintDatFile.out to see if your passengers3.dat is correct.

4. The term "const" must be used WHEREVER possible in the prototypes of functions. Note that const is not needed for any value passed by value.
5. No #defines are allowed except those used for header guards. All constants, other than 0 and 1, should be declared as static const "variables," or identifiers in an enum. Those constants used in classes, should be declared within their respective classes. An enum statement at the top of main.cpp should be used to assign the values for the constants used in the switch statement in main(). There is no longer a need to #define `TRUE` since C++ has the built-in constant `true`.
6. Code must be submitted by exactly one member of each team. Double submissions, or errors in the authors.csv format will result in the team losing five points. Your handin command line will be:
   handin cs40a p4 authors.csv main.cpp plane.h plane.cpp flight.h flight.cpp flights.h flights.cpp passenger.h passenger.cpp utilities.h utilities.cpp Makefile

Suggestions:

Because there are fairly dramatic changes in the program, you should try to address as small a chunk as possible at a time. Compile whenever possible, and test run whenever possible. The best way to minimize the size of chunks to be newly tested is to work from reading to writing to changing. Each time you modify a function make sure it compiles before moving on to another. Write the comments as you go. They are there to help you understand the code. Here is a possible order that should compile, and run without error (albeit incompletely) after each major step.

Create the Flights class, and change the flights array to a Flights class object in existing functions.

1. Write Flights class including a default constructor that initializes the flights capacity to 2 Flight*.
2. Modify main() to use a Flights object, and move all functions that took a Flight array to the Flights class..
   2.1. Modify each function call in main(), except getChoice() to use a Flights object to call a Flights method. There is no need to pass numFlights to the functions since Flights has the size variable.
   2.2. Move all the functions called from main(), except getChoice() to the Flights class. writeFlights() and freeFlights() should be combined to form the destructor. The destructor need only store the size in flights2.csv, and then close the file. addFlights() and readFlights() will call the Flights::insert() method.
   2.3. Write Flights::insert() that must resize the array as needed, and maintain the order of the array based on the flightNums by shifting larger flightNums away from the zeroth index to make room for a new, smaller Flight*.

Modify Flight class methods to handle flights.csv

3. Since the Plane class will usually be dealing with passengers2.dat, comment out <u>all</u> lines that mention plane in <u>all</u> of the Flight methods.
4. Modify Flight::readFlight() to account for the new format. For now, ignore() the rows and width at the end of each line. Set the number of flights properly in Flight::readFlights(). You should now be able to use the addPassenger menu item to look at the Flight info.
5. Move the Flight::writeFlight() to the Flight destructor, and account for the new format. The destructor should open flights2.csv to append the information (with dummy rows and width for now), and then close the file. One trick for testing would be to rename flights2.csv to flights.csv (after saving the original flights.csv), and see if it looks correct with addPassenger.

Modify Plane class methods to handle flights.csv information and a passenger array of int**.

6. To allow smaller steps in the compilation testing use /* and */ to comment out all of the content of the Plane methods except getRow(), and showGrid().
7. Modify the Plane class to have passengers array of int**.
8. In Flight::readFlight(), uncomment the call to the Plane constructor, and eliminate the ignore(). Pass both the flightNum and inf to the Plane constructor. Modify the Plane constructor to account for the format of flights.csv. Delete from the constructor the loop that reads information about passengers. Modify the loops in the constructor that create the passengers array so that they only create the passengers array of ints, and initialize the elements to -1.

9. Modify Plane::showGrid() so that it relies on -1 instead of NULL for an empty seat.
10. Uncomment and modify ~Plane() so that it only removes the passengers array as it is now constituted.
11. In ~Flight() uncomment the statement that deletes the Plane.

Create Passenger class, and modify functions to handle passengers.dat, passengers2.dat, and passengers3.dat

12. Passengers class. Write the data section EXACTLY as specified. There need be no functions yet. Since only the Plane class will interact with the passenger.dat file, and thus the Passenger class objects, it is reasonable to make the Plane class a friend of the Passenger class. You will need to #include "plane.h" at the top of passenger.h to do this.
13. In the Passenger class, write a public static member function named copyPassengers() that copies passengers.dat to passengers2.dat. This function should be called at the beginning of main() before readFlights(). Use diff, or PrintDatFile.out to check that your passengers2.dat matches passengers.dat.
14. Modify the Plane constructor to read through the passengers2.dat file, one Passenger at a time. If a passenger is on the Plane's flight, determine the byte position of that passenger in the file, and store that position in the appropriate passenger array element, and increment reserved. You will be using read(), and tellg().
15. Uncomment the call to Plane::addPassenger in Flight::addPassenger(), and pass the flightNum to it. Uncomment the body of Plane::addPassenger() up to, and including, the call to showGrid(). Add " return true;" after the showGrid(). Your program should run, and the addPassenger menu item should now show the correct grid for each flight.
16. Write a default constructor, and a "standard" constructor for the Passenger class. A "standard" constructor has a parameter for each data member of the class.
17. Uncomment the rest of Plane::addPassenger(), and eliminate the storage of the name code as well as the return statement you added in step 14. Modify Plane::addPassenger to append the passenger information to passengers2.dat, and set the associated passengers array element to the byte position in the file. You will be using write(), ios::app, and tellp(). Adding the passenger should work perfectly now. You test this by looking at the grid and passengers2.dat.
18. Modify ~Flight to call Plane::writePlane() before it deletes the plane. ~Flight should no longer write bogus rows and columns to flights2.csv. writePlane() now takes a flightNum as well as an ofstream as its parameters. Uncomment writePlane(). Besides writing to flights.csv, writePlane() will now open passengers2.dat and append the Plane's passengers to passengers3.dat based on the positions specified in the passengers array. You will be using read(), write(), ios::app, and seekg(). Use PrintDatFile.out to check your passengers3.dat.

Add the three new menu items.

19. Add an enum at the top of main.cpp that holds a capitalized identifier for each menu item.
20. Write addFlight() in the Flights, Flight, and Plane classes, create a switch statement in main(), and add a menu item. You need to add a default Plane constructor that does nothing.
21. Write removePassenger() methods in Flights, Flight, and Plane classes, add an item to the main(). Remember to set the flightNum of the passenger to -1 in passengers2.dat, which will take an fstream opened with the `ios::binary | ios::in | ios::out` mode, and the use of seekp() and seekg().
22. Write removeFlight() in the Flights, Flight, and Plane classes, add an item to the main() switch, and add a menu item. You will need to go through the whole passengers array, and set the flightNum of the passengers to -1 in passengers2.dat.

Change the #defines into consts, and move the constants into the class declarations

23. Change all of the #defines of constants into static const. Move the declarations of constants in the class files into their declarations in their respective header files.

```
[ssdavis@lect1 p4]$ CreateFlights.out
Seed: 9
Enter number of flights: 5
[ssdavis@lect1 p4]$ cat flights.csv
5
815,Sacramento,Stockton,4,4
583,Reno,Sacramento,7,4
265,Reno,Los Angeles,9,2
201,Reno,San Francisco,7,2
552,Stockton,Reno,8,6
[ssdavis@lect1 p4]$
```

```
ECS Flight Reservation Menu
0. Exit.
1. Add Passenger.
2. Remove Passenger.
3. Add Flight.
4. Remove Flight.
```

```
[ssdavis@lect1 p4]$ cat BigTest.txt
1
201
Flintstone,Fred
7
B
1
201
Mays,Willie
6
B
1
201
2
583
Dobbs,Alex
3
553
Davis
San Diego
3
8
1
553
Fields,W.C.
2
H
4
265
1
553
Marx,Groucho
1
G
0

[ssdavis@lect1 p4]$

[ssdavis@lect1 p4]$ airline.out < BigTest.txt
```

```
[ssdavis@lect1 p4]$ PrintDatFile.out
Please enter the name of file you wish printed: passengers.dat
583 4 D Hashemzadeh,Victor
552 4 D Ahn,Aaron
583 1 B Fong,Clare
815 1 A Becker,Pao-Hsiang
552 6 F Balog,Ryan
201 3 B Li,Peter
815 4 B Li,Peter
552 4 B Law,Edward
552 1 B Mak,Daniel
201 7 A Steiner,Gail
583 4 C Brown,Jason
815 4 C Nakodary,Michael
552 3 A Vasquez,Daniel
552 1 D Kuffel,Frank
815 3 D El-Sissi,Brian
201 4 B Yuan,Edward
815 1 C Lozano,Way
201 1 B Inocencio,Carlton
265 7 A Chinn,Raj
201 3 A Parcher,Christopher
265 2 A Tran,Rex
552 2 C Aroyan,Eddie
201 5 A Wan,Amir
552 3 B Loden,Hai
265 1 A Lau,Evelyn
552 5 D Ho,Jane
265 3 B Aguirre,Aaron
552 6 D Ballelos,Ernie
583 7 C Nguyen,Jesse
583 2 C Viswanathan,Long
552 1 E Nguyen,Travis
583 5 B Martin,Rochak
552 8 B Wei,Kwasi
552 5 A Liu,Anthony
552 6 C Rood,Shuka
583 5 D Dobbs,Alex
265 2 B Morgan,Tri
552 5 E Genitiano,David
552 6 A Bleier,Wei-Chi
552 7 D Novikov,Artem
201 6 A Wang,Zhen
583 7 A Huynh,Jeffrey
201 2 B Lock,Carlos
583 3 B Shelley,Jason
201 2 A Lee,James
552 3 C Lee,Mandeep
552 3 D Nakodary,Michael
201 4 A Kong,Alexander
583 1 C Guerin,Justin
265 8 A Li,Alan
265 3 A Liu,Anthony
201 1 A Ahmadi,Juan
552 6 B Chinn,Hoang
552 7 B Molano,Laurent
201 5 B Wang,Kenneth
815 1 D Kee,Albert
265 5 B Wang,Jordan
552 5 C Uthus,Nicholas
265 4 B Martin,Rochak
552 5 B Hong,Gloria
583 1 D Genitiano,David
265 6 A Guan,Larry
265 7 B Lam,Stephanie
583 2 A Ye,Eric
552 8 D Ngo,Tae-Yoon
815 2 A Abed-Amoli,Sukhbir
815 3 B Dham,Andrew
583 5 A Alabanza,Alison
583 2 B Huynh,Zane
265 8 B Chau,Brandon
[ssdavis@lect1 p4]$
```

```
[ssdavis@lect1 p4]$ PrintDatFile.out
Please enter the name of file you wish printed: passengers2.dat
583 4 D Hashemzadeh,Victor
552 4 D Ahn,Aaron
583 1 B Fong,Clare
815 1 A Becker,Pao-Hsiang
552 6 F Balog,Ryan
201 3 B Li,Peter
815 4 B Li,Peter
552 4 B Law,Edward
552 1 B Mak,Daniel
201 7 A Steiner,Gail
583 4 C Brown,Jason
815 4 C Nakodary,Michael
552 3 A Vasquez,Daniel
552 1 D Kuffel,Frank
815 3 D El-Sissi,Brian
201 4 B Yuan,Edward
815 1 C Lozano,Way
201 1 B Inocencio,Carlton
-1 7 A Chinn,Raj
201 3 A Parcher,Christopher
-1 2 A Tran,Rex
552 2 C Aroyan,Eddie
201 5 A Wan,Amir
552 3 B Loden,Hai
-1 1 A Lau,Evelyn
552 5 D Ho,Jane
-1 3 B Aguirre,Aaron
552 6 D Ballelos,Ernie
583 7 C Nguyen,Jesse
583 2 C Viswanathan,Long
552 1 E Nguyen,Travis
583 5 B Martin,Rochak
552 8 B Wei,Kwasi
552 5 A Liu,Anthony
552 6 C Rood,Shuka
-1 5 D Dobbs,Alex
-1 2 B Morgan,Tri
552 5 E Genitiano,David
552 6 A Bleier,Wei-Chi
552 7 D Novikov,Artem
201 6 A Wang,Zhen
583 7 A Huynh,Jeffrey
201 2 B Lock,Carlos
583 3 B Shelley,Jason
201 2 A Lee,James
552 3 C Lee,Mandeep
552 3 D Nakodary,Michael
201 4 A Kong,Alexander
583 1 C Guerin,Justin
-1 8 A Li,Alan
-1 3 A Liu,Anthony
201 1 A Ahmadi,Juan
552 6 B Chinn,Hoang
552 7 B Molano,Laurent
201 5 B Wang,Kenneth
815 1 D Kee,Albert
-1 5 B Wang,Jordan
552 5 C Uthus,Nicholas
-1 4 B Martin,Rochak
552 5 B Hong,Gloria
583 1 D Genitiano,David
-1 6 A Guan,Larry
-1 7 B Lam,Stephanie
583 2 A Ye,Eric
552 8 D Ngo,Tae-Yoon
815 2 A Abed-Amoli,Sukhbir
815 3 B Dham,Andrew
583 5 A Alabanza,Alison
583 2 B Huynh,Zane
-1 8 B Chau,Brandon
201 7 B Flintstone,Fred
201 6 B Mays,Willie
553 2 H Fields,W.C.
553 1 G Marx,Groucho
[ssdavis@lect1 p4]$
```

```
[ssdavis@lect1 p4]$ PrintDatFile.out
Please enter the name of file you wish printed: passengers3.dat
201 1 A Ahmadi,Juan
201 1 B Inocencio,Carlton
201 2 A Lee,James
201 2 B Lock,Carlos
201 3 A Parcher,Christopher
201 3 B Li,Peter
201 4 A Kong,Alexander
201 4 B Yuan,Edward
201 5 A Wan,Amir
201 5 B Wang,Kenneth
201 6 A Wang,Zhen
201 6 B Mays,Willie
201 7 A Steiner,Gail
201 7 B Flintstone,Fred
552 1 B Mak,Daniel
552 1 D Kuffel,Frank
552 1 E Nguyen,Travis
552 2 C Aroyan,Eddie
552 3 A Vasquez,Daniel
552 3 B Loden,Hai
552 3 C Lee,Mandeep
552 3 D Nakodary,Michael
552 4 B Law,Edward
552 4 D Ahn,Aaron
552 5 A Liu,Anthony
552 5 B Hong,Gloria
552 5 C Uthus,Nicholas
552 5 D Ho,Jane
552 5 E Genitiano,David
552 6 A Bleier,Wei-Chi
552 6 B Chinn,Hoang
552 6 C Rood,Shuka
552 6 D Ballelos,Ernie
552 6 F Balog,Ryan
552 7 B Molano,Laurent
552 7 D Novikov,Artem
552 8 B Wei,Kwasi
552 8 D Ngo,Tae-Yoon
553 1 G Marx,Groucho
553 2 H Fields,W.C.
583 1 B Fong,Clare
583 1 C Guerin,Justin
583 1 D Genitiano,David
583 2 A Ye,Eric
583 2 B Huynh,Zane
583 2 C Viswanathan,Long
583 3 B Shelley,Jason
583 4 C Brown,Jason
583 4 D Hashemzadeh,Victor
583 5 A Alabanza,Alison
583 5 B Martin,Rochak
583 7 A Huynh,Jeffrey
583 7 C Nguyen,Jesse
815 1 A Becker,Pao-Hsiang
815 1 C Lozano,Way
815 1 D Kee,Albert
815 2 A Abed-Amoli,Sukhbir
815 3 B Dham,Andrew
815 3 D El-Sissi,Brian
815 4 B Li,Peter
815 4 C Nakodary,Michael
[ssdavis@lect1 p4]$
[ssdavis@lect1 p4]$ cat flights2.csv
5
201,Reno,San Francisco,7,2
552,Stockton,Reno,8,6
553,Davis,San Diego,3,8
583,Reno,Sacramento,7,4
815,Sacramento,Stockton,4,4
[ssdavis@lect1 p4]$
```