New concepts: None, this is just a review of C. Executable name: airline.out
File names: main.cpp, flight.h, flight.cpp, plane.h, plane.cpp, utilities.h, utilities.cpp, Makefile, authors.csv.
Due: Wednesday, April 13th at 11:59 PM in p1 of cs40a using handin.

Format of authors.csv: author1_email,author1_last_name,author1_first_name
author2_email,author2_last_name,author2_first_name
For example:
simpson@ucdavis.edu,Simpson,Homer
potter@ucdavis.edu,Potter,Harry

This will be the first of a series of programming assignments to create an airline reservation system. For this program, you will write a program to read in the information from the file reservations.txt, add passengers to specified flights, and then write a new file, reservations2.txt that contains all of the updated information. The information should be stored in an array of typedef Flight structs (see below). Since the number of flights is not known at the beginning, you will have to use malloc to allocate the array.

The reservations.txt file is somewhat complex. The first line indicates how many flights there are. There then is a series of lines for each flight. For each flight the lines will be: 1) flight number; 2) Origin city; 3) destination city; 4) number of rows, width of rows, and number of seats already reserved; and lines 5 on) row number, seat letter, and name of each passenger. Each piece of information should be stored in a separate variable. The rows, width, reserved, and information about the passengers will be stored in a typedef Plane struct (see below). Since the length of passenger names varies so much, you will use a two-dimensional array of char pointers that is completely dynamically allocated.

While your program may assume the reservations.txt file, names, and seat letters will be in a valid format, the user numerical entries may be invalid in any way possible. Your program must be robust enough to handle ANY user input for numerical entries. It must range check all numerical values. Leading and/or trailing white space is not sufficient cause to declare an input invalid. Please be aware that the examples of error checking on this sheet do not explore all the erroneous format possibilities. It is up to you to develop your own tests. The testing script provided later this week will NOT explore all of the possibilities that the real grading script will use. Names will be no longer than 80 chars, including the '\0'.

Specifications:
1. main.cpp
   1.1. main() should be at a high level of abstraction. It may only contain: a declaration of Flight* named flights, a declaration of an int named numFlights, function calls, one while loop, and a return statement.
      1.1.1. Though my main() contains only six semi-colons, i.e. statements, yours may contain up to eight semi-colons.
   1.2. No element of a Flight struct (e.g. origin), nor Plane struct (e.g. rows) may be mentioned in main.cpp. Note that the design checker will look for each variable of the structs in your main.cpp. So do not use such identifiers (e.g. flightNum) as local variables, in main.cpp.
2. All dynamically allocated memory must be freed in functions dedicated to that task before the return statement of main() is reached.
3. Function Locations
   3.1. Because you will be adapting this program to C++ in program #3, almost all of your functions should have either a Plane* or a Flight*, but not both, as one of their parameters.
   3.2. Those functions that have a Plane* as a parameter must have their prototypes in plane.h, and their implementation code in plane.cpp. Only such functions may access the elements of the Plane struct.
   3.3. Those functions that deal with one flight should have a Flight* as a parameter, and must be written in flight.h, and flight.cpp. Only such functions may access the elements of the Flight struct.
   3.4. Those functions that deal with the entire Flight array, should be in main.cpp.
   3.5. Functions that are used by more than one .cpp file should be placed in utilities.h, and utilities.cpp.
      3.5.1. You should write one function, named getNumber() that returns an int, and reads each needed number from the user. This one function will ensure that your program can handle any input for numbers, and avoids duplicated code in multiple files. Though atoi() would be tempting, it will not help you here, but .
4. All numerical constants, other than 0 and 1, must be declared in #defines. Note that the definition of the Flight struct must be changed to fulfill this requirement.
5. The style of your program must obey the programming style specified in the style handout. Remember the body of any function cannot be more than 35 lines long.
6. If you wish to write the program in C++ you may, though you must still use structs instead of classes.
7. The output of your program must be identical to that of mine. Your reservations2.txt must also be identical to mine. You will find my executable, and reservations.txt in ~ssdavis/40/p1.
8. 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 p1 authors.csv main.cpp plane.h plane.cpp flight.h flight.cpp utilities.h utilities.cpp Makefile

Suggestions:

1. Use iterative top design with function stubs.
    1.1. Write main() and the stubs for the functions it calls, and getting it to compile without warnings before writing anything else. For those stubs that return values, have them return values that will prevent infinite loops.
    1.2. You will continue to write one function at a time, adding stubs as needed, and compile it until there are no warnings before writing another function. The following is an ordering that will allow testing after each step.
        1.2.1. Write the function to read the first line of the file, allocate the Flight array, and call a function to read a Flight.
        1.2.2. Write the function to read the Flight ints, allocate the Plane, and call a function to read a Plane.
        1.2.3. Write the function to read a Plane's information.
        1.2.4. Write the function to deal with the menu.
        1.2.5. Write getNumber(). You will find getchar() and the ctype.h library very useful for this function.
    1.3. Continue in the same manner to deal with the different steps involved in the add Passenger routines.
    1.4. Continue in the same manner to deal with the different steps to write reservations2.txt. Since these steps follow the pattern of reading from reservations.txt so closely you may wish to just copy your reading functions, and modify them to write instead of read.
    1.5. Continue in the same manner to deal with the different steps to free the dynamic memory. These steps also follow the same pattern as the reading from reservations.txt, but are even simpler.
2. By initializing all of the rows * width individual positions in the passengers two-dimensional array of the Plane structs with NULL, you can easily create the reserved seat grid displayed by the Add Passenger routine.
3. To remove the '\n' and/or '\r' read with fgets(), use strtok(). By combining both chars in strtok(), you allow your code to work in any operating system even though the operating systems use different end-of-line character(s).
4. Always place the #includes of standard header files above any #include of your own header files. This minimizes the chances that compiler will confuse you by reporting errors in the standard headers when they are actually errors in your header files.
5. **START EARLY!!!**

```
typedef struct
{
  int flightNum;
  char origin[20];
  char destination[20];
  Plane *plane;
} Flight;
```

```
typedef struct
{
  int rows;
  int width;
  int reserved;
  char ***passengers; // 2-dimensional
} Plane;
```

```
[ssdavis@lect1 p1]$ cat reservations.txt
4
230
Reno
Los Angeles
8 2 10
8A Gary Lam
5A Harnit Prado
1A Pele Chan
8B David Shah
2A Ryan Chock
7B Daniel Mak
4B Sze-Kin Barahona
2B Danny Tran
5B Tri Chiong
3A Aren Chuang
463
Stockton
Los Angeles
8 2 14
5B Arjun Riordan
5A Catherine Gawthorpe
3A Eric Brown
6B Tony Iwahashi
6A Joanne Lam
7B Erich Helmhold
2A Mikhail Ip
1A Dinh-Tuong Leung
4A David Wong
1B Vincent Ho
2B Angela Chang
```

```
3B Dernie Tran
8B Man Li Horng
7A William Chaves
770
Sacramento
San Francisco
6 2 10
5A Chukwuemeka Mansouri
5B Hwai-En Eng
3A Victor Gunel
2A Sze-Kin Barahona
3B Sylvia Dinh
6A Cary Tong
1A Thaya Hung
6B Way Lozano
1B Jimmy Sudame
4A David Shah
221
Reno
Davis
7 4 8
3C Lawrence Heinz
7D Men-Jyn Maung
2B Melvin Begusch
4D Alison Alabanza
6B Ryan Balog
3D Michael Park
1A Chi Ho Tung
6C Christine Law
[ssdavis@lect1 p1]$
```

```
[ssdavis@lect1 p1]$ airline.out
```

```
ECS Flight Reservation Menu
0. Exit
1. Add Passenger.

Please enter your choice: 2a
Your number is invalid.
Please try again.

Please enter your choice: 2
2 is not an available choice.
Please try again.

Please enter your choice: 1
Flt# Origin                         Destination
230  Reno                           Los Angeles
463  Stockton                       Los Angeles
770  Sacramento                     San Francisco
221  Reno                           Davis

Flight number (0 = exit): 4632
We do not have a flight number 4632.
Please try again.

Flight number (0 = exit): 463
Please enter the name of the passenger: George Bush
Row# AB
 1   XX
 2   XX
 3   XX
 4   X-
 5   XX
 6   XX
 7   XX
 8   -X

X = reserved.

Please enter the row of the seat you wish to reserve: 9
There is no row #9 on this flight.
Please try again.

Please enter the row of the seat you wish to reserve: 4
Please enter the seat letter you wish to reserve on row #4: A
That seat is already occupied.
Please try again.

Please enter the row of the seat you wish to reserve: 4
Please enter the seat letter you wish to reserve on row #4: B
We have reserved seat 4B on flight 463 for George Bush.

ECS Flight Reservation Menu
0. Exit
1. Add Passenger.

Please enter your choice: 1
Flt# Origin                         Destination
230  Reno                           Los Angeles
463  Stockton                       Los Angeles
770  Sacramento                     San Francisco
221  Reno                           Davis

Flight number (0 = exit): 463
Please enter the name of the passenger: Bill Clinton
Row# AB
 1   XX
 2   XX
 3   XX
 4   XX
 5   XX
 6   XX
```

```
 7   XX
 8   -X

X = reserved.

Please enter the row of the seat you wish to reserve: 8
Please enter the seat letter you wish to reserve on row #8: A
We have reserved seat 8A on flight 463 for Bill Clinton.

ECS Flight Reservation Menu
0. Exit
1. Add Passenger.

Please enter your choice: 1
Flt# Origin                          Destination
230  Reno                            Los Angeles
463  Stockton                        Los Angeles
770  Sacramento                      San Francisco
221  Reno                            Davis

Flight number (0 = exit): 463
We are sorry but Flight #463 is full.

ECS Flight Reservation Menu
0. Exit
1. Add Passenger.

Please enter your choice: 0
Goodbye.
[ssdavis@lect1 p1]$ cat reservations2.txt
4
230
Reno
Los Angeles
8 2 10
1A Pele Chan
2A Ryan Chock
2B Danny Tran
3A Aren Chuang
4B Sze-Kin Barahona
5A Harnit Prado
5B Tri Chiong
7B Daniel Mak
8A Gary Lam
8B David Shah
463
Stockton
Los Angeles
8 2 16
1A Dinh-Tuong Leung
1B Vincent Ho
2A Mikhail Ip
2B Angela Chang
3A Eric Brown
3B Dernie Tran
4A David Wong
4B George Bush
5A Catherine Gawthorpe
5B Arjun Riordan
6A Joanne Lam
6B Tony Iwahashi
7A William Chaves
7B Erich Helmhold
8A Bill Clinton
8B Man Li Horng
770
Sacramento
San Francisco
6 2 10
1A Thaya Hung
1B Jimmy Sudame
2A Sze-Kin Barahona
3A Victor Gunel
3B Sylvia Dinh
4A David Shah
5A Chukwuemeka Mansouri
5B Hwai-En Eng
6A Cary Tong
6B Way Lozano
221
Reno
Davis
7 4 8
1A Chi Ho Tung
2B Melvin Begusch
3C Lawrence Heinz
3D Michael Park
4D Alison Alabanza
6B Ryan Balog
6C Christine Law
7D Men-Jyn Maung
[ssdavis@lect1 p1]$
```