ECS 40 **Program #4 (50 points)** **Spring 2015**

Due: Wednesday, May 6th at 11:59pm using handin to p4 directory of cs40a.
New concepts: unformatted files, stream manipulation, overloaded operator. Name of executable: simulator.out
File names: main.cpp, city.cpp, city.h, plane.cpp, plane.h, vector.cpp, vector.h, Ins.c, Makefile-Debug.mk, and authors.txt.

For this assignment you will learn to use the gdb GUI, ddd, and the Netbeans IDE (Integrated Development Environment).

ddd Tutorial (10 points)

Follow the directions of the DDD tutorial available online at http://heather.cs.ucdavis.edu/~matloff/Debug/Debug.pdf. Note that Professor Matloff's tutorial stops short of completely debugging Ins.c. You must finish the job on your own. You will find Ins.c in ~ssdavis/40/p4. When done completely debugging Ins.c, handin it.

There are at least four ways to gain access to ddd:

1. Go to the basement of Kemper and select DDD from the Programming/More Programming Tools menu.
2. To use ddd at home under Windows, on programs developed at home.
   - 2.1. Install cygwin (available for free from cygwin.com) with g++, openssh, and ddd. The selection of ddd should automatically install the X windowing server for cygwin.
   - 2.2. Once cygwin is installed, type **xinit&** at the cygwin command prompt to open an X window, and then type **ddd** at the prompt.
3. To use ddd at home under Windows, on programs developed in the CSIF.
   - 3.1. Install cygwin with at least the X server (I would still suggest installing g++ and ddd).
   - 3.2. Once cygwin is installed, type **xinit&** at the cygwin command prompt to open an X window.
   - 3.3. Type ssh –X username@CSIF_computername
   - 3.4. Once you have logged into the CSIF computer, change to the appropriate directory, and then type **ddd&**
4. To use ddd at home under MacIntosh OS X, on programs developed in the CSIF
   - 4.1. Open an X term. (See the MacIntosh help to install the X package)
   - 4.2. Type **ssh –X username@CSIF_computername**
   - 4.3. Once you have logged into the CSIF computer, change to the appropriate directory, and then type **ddd&**

Netbeans (10 points)

Netbeans is installed on the CSIF computers under Applications->Programming->Netbeans, or just Searching for Netbeans. You can download your own free copy from netbeans.org. From the Download page choose the "C/C++" version. There are many tutorials available from the Netbeans start page or directly at http://www.netbeans.org/kb/trails/cnd.html. It is up to you to choose what tutorials you need.

To match the indentation requirement of 2 spaces, within NetBeans you will first need to create a C++ project. Once you are working with a C++ project, go to Tools->Options->Editor->Formatting and set the "Number of Spaces per Indent" and "Tab Size" to 2. The "Expand Tabs to Spaces" should already be checked.

To prove that you used NetBeans, we want you to handin Makefile-Debug.mk for your simulator project. This worth 10 points of the 50 points for the assignment. To have Makefile-Debug.mk created properly, you must create a C/C++ Application Project in Netbeans (don't select C/C++ Project with Existing Sources). Makefile-Debug.mk is located in the nbproject directory of the directory NetBeans will create for your funix project. Make sure it's there after creating your project. Even though NetBeans creates its own Makefile for your project, we will be testing with the Makefile provided in ~ssdavis/40/p4. You should make sure that your program also compiles without warnings using that Makefile!!!

simulator.out (30 points, 2 hours)

For this assignment, you will incorporate information about the characteristics of planes to determine the best way to transport the passengers from a city. You will be adding a new class, Plane, that has its information stored in an unformatted file, named planes.dat. Unlike previous assignments, there is no walkthrough of development. It is time for you to apply what you have learned about incremental development including function stubs, and incremental testing.

You may use either your own p3 code, or my p3 code from ~ssdavis/40/p3 (available Thursday morning) as your starting point for the assignment. There is no problem with plagiarism if you use my code. You will find my executable, and the three data files in ~ssdavis/40/p4. As usual, the format and values of your program must match mine.

Further specifications:

1. main.cpp
   1.1. There will now be new functions to handle the three new menu items: displayPlaneInformation(), addPlaneInformation(), and determineBestPlane().
2. Plane class
   2.1. To properly access planes.dat you MUST declare your class Plane with the following data members in this order:

```
char name[12];
int passengers;
int range; // in miles
int speed; // in mile per hour
int fuel;  // in U.S. gallons
int price;  // in U.S. dollars
```

   2.2. All of the data in the files I have provided is accurate. We will reduce by half the range and fuel capacity of all of the aircraft. planes.dat already has the range and fuel reduced in half.
   2.3. In main(), you should declare an array of 10 Planes and an int to keep track of the number of Planes in the array. Note that there is no need to dynamically allocate the array.
   2.4. Displaying a plane should use an overloaded insertion operator.
      2.4.1. In displaying the information for a plane, you may not use spaces in any constant strings. You must use setw() instead.
      2.4.2. The price of a gallon of jet fuel is $3.39 which must be set as a static const double in the Plane class.
   2.5. Adding a plane
      2.5.1. When adding a plane to planes.dat, you should append to the file rather than rewriting the whole file.
      2.5.2. You may assume that the user will enter proper data for a plane.
      2.5.3. You should add only one method to the Plane class to implement the data input.
3. Determining best planes for a route.
   3.1. Obviously a plane must have the range to make the trip between the cities.
   3.2. Hint: Since the Vector and City calcDistance() methods already calculate distance and the number of passengers, you need just provide them a few new parameters to have access to that data. This is a better design than having a complex formula duplicated.
   3.3. The cost of trip is the sum of the fuel costs, flight attendant salaries, pilot salaries, and maintenance cost using doubles with total rounded up. You will find ceil() useful. I suggest you calculate each separately.
      3.3.1. Fuel costs would be based on miles per gallon from the plane's range and fuel capacity.
      3.3.2. There is one attendant for every 100 passengers possible, rounded up. They are paid $30 per hour. They work 2 hours on the ground plus the flight time, rounded up to whole hours.
      3.3.3. There are two pilots for every plane. They are paid $100 per hour. They work 2 hours on the ground plus the flight time, rounded up to whole hours.
      3.3.4. Maintenance cost are 0.0025% * aircraft price * hours.
   3.4. For this program you should list the plane with the lowest total cost to deliver <u>all</u> of the passengers between the two cities.
   3.5. You should rely on Plane::getName() to provide the name of the best plane.
4. General programming standards beyond those provided on the Programming Standards handout.
   4.1. main() may only contain a variable declarations, function calls, and a return statement.
   4.2. All functions must be implemented in .cpp files.
   4.3. Use const wherever possible in function parameters. The const checker will be administered only <u>after</u> the deadline.
   4.4. All reference parameters should be const. If they cannot be const, then you should be using pointers.

Boeing Airliner Specifications

Name	Passengers	Fuel	Fuel/2	Range	Range/2	Speed	Price x 10^6	MPG	G/Pass-Mile
737-800	189	6875	3437	3060	1530	524	$67.8	0.445	0.0119
747-600	524	57285	28642	7260	3630	567	$216	0.127	0.0151
767-300ER	350	24080	12040	5975	2987	530	$134	0.248	0.0115
777-300	550	45220	22610	6015	3007	560	$212	0.133	0.0136
787-3	330	11086	5543	3050	1525	587	$130	0.275	0.0110

```
[ssdavis@lect1 p4]$ simulator.out

Flight Simulator Menu
0. Done.
1. Determine distance and passengers between two airports.
2. Determine all traffic from one airport.
3. Display planes information.
4. Add plane information.
5. Determine best plane between two airports.

Your choice (0 - 5): 3

Plane Information
Name        Pass. Range Speed Fuel   MPG   $/mi  Price * 10^6
737-800     189   1,530 524    3,437 0.445  7.62 $ 67.8
747-600     524   3,630 567   28,642 0.127 26.75 $216.0
767-300ER   350   2,987 530   12,040 0.248 13.66 $134.0
777-300     550   3,007 560   22,610 0.133 25.49 $212.0

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 5

Please enter two airport abbreviations (XXX XXX): SFO NYC
Passengers Miles  Trips Name      Cost
2531       2570   8     767-300ER $428816

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 5

Please enter two airport abbreviations (XXX XXX): SFO SEA
Passengers Miles  Trips Name      Cost
181        679    1     737-800   $8408

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 4
Name: 787-3
Passengers: 330
Fuel capacity (in U.S. gallons): 5543
Range (in miles): 1525
Speed (in mph): 587
Price: 130000000

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 3

Plane Information
Name        Pass. Range Speed Fuel   MPG   $/mi  Price * 10^6
737-800     189   1,530 524    3,437 0.445  7.62 $ 67.8
747-600     524   3,630 567   28,642 0.127 26.75 $216.0
767-300ER   350   2,987 530   12,040 0.248 13.66 $134.0
777-300     550   3,007 560   22,610 0.133 25.49 $212.0
787-3       330   1,525 587    5,543 0.275 12.32 $130.0

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 0
[ssdavis@lect1 p4]$ simulator.out
```

```
[ssdavis@lect1 p4]$ simulator.out

Flight Simulator Menu
0. Done.
1. Determine distance and passengers between two airports.
2. Determine all traffic from one airport.
3. Display planes information.
4. Add plane information.
5. Determine best plane between two airports.

Your choice (0 - 5): 3

Plane Information
Name        Pass. Range Speed Fuel   MPG   $/mi  Price * 10^6
737-800     189   1,530 524    3,437 0.445  7.62 $ 67.8
747-600     524   3,630 567   28,642 0.127 26.75 $216.0
767-300ER   350   2,987 530   12,040 0.248 13.66 $134.0
777-300     550   3,007 560   22,610 0.133 25.49 $212.0
787-3       330   1,525 587    5,543 0.275 12.32 $130.0

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 0
[ssdavis@lect1 p4]$
[ssdavis@lect1 p4]$ mv planes.dat planes2.dat
[ssdavis@lect1 p4]$ simulator.out

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 4
Name: 737-800
Passengers: 189
Fuel capacity (in U.S. gallons): 3437
Range (in miles): 1530
Speed (in mph): 524
Price: 67800000

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 3

Plane Information
Name        Pass. Range Speed Fuel   MPG   $/mi  Price * 10^6
737-800     189   1,530 524    3,437 0.445  7.62 $ 67.8

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 5

Please enter two airport abbreviations (XXX XXX): SFO LAX
Passengers Miles  Trips Name      Cost
1173       339    7     737-800   $31208

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 5

Please enter two airport abbreviations (XXX XXX): LAX NYC
No planes available

Flight Simulator Menu
// Edited by Sean to save space
Your choice (0 - 5): 0
[ssdavis@lect1 p4]$
```