Due: Wednesday, May 20th at 11:59pm using handin to p6 directory of cs40a.
New concepts: inheritance, templates, recursion, polymorphism Name of executable: simulator.out
File names: Makefile, main.cpp, city.cpp, city.h, plane.cpp, plane.h, list.cpp, list.h, route.cpp, route.h airport.cpp, airport.h, shapes.cpp, and authors.txt.

Simulator.out (1.5 hours, 40 points)

For this assignment, you will add airline flights to the cities, and the ability to determine the best plane to use for a route between two airports that involves passing through multiple airports. To accomplish this functionality you will need to derive a Route class from the Plane class that has additional information about the airports that the plane visits between its origin and destination. You will need an Airport class that is derived from the City class that has additional information about the flights of seven airlines that leave from each city. You will also be converting all non-char arrays to a general, sorted templated list.

You may use either your own p5 code, or my p5 code from ~ssdavis/40/p5/SeansSrc (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/p6. As usual, the format and values of your program must match mine.

Further specifications and development guidelines:

1. Your program should compile and run perfectly at the end of each major step.
2. Convert CityList and CityNode to template classes List and ListNode. (10 minutes)
   - 2.1. This is fairly straightforward. Here are some hints.
   - 2.2. Copy citylist.h to list.h and citylist.cpp to list.cpp, and then use four global search and replaces will get you most of the way there.
     - 2.2.1. The three lines that declare classes in list.h do not have the <T>. However T is needed for parameters, data members, and return types.
     - 2.2.2. The <T> is needed to differentiate the class when using the scope operator in list.cpp.
     - 2.2.3. To make the List class more generic, change the ListNode data member's name from city to data.
     - 2.2.4. Add a #include "list.cpp" to the end of list.h.
   - 2.3. Remove any reference to citylist.o in your Makefile. Wherever citylist.h is mentioned in a dependency list in your Makefile, change it to list.h and follow it with list.cpp. That way if list.cpp is changed, the dependent object file will be recompiled.
   - 2.4. Adjust the class names to List<T> and ListNode<T> where appropriate in list.cpp. Note that the names of functions never have a <T>, only the names of classes.
   - 2.5. In main.cpp, a global search and replace will again address most of the necessary changes.
3. Make the city list sorted by population from largest to smallest. (7 minutes)
   - 3.1. Sort utilizing the < operator of the data in += of List.
   - 3.2. You should remove the tail pointer from the List class.
   - 3.3. You will need to add an overloaded < operator to City.
4. Make the Planes array a List <Plane> that is sorted by name. (10 minutes)
   - 4.1. You will need to add an operator< to the Plane class that compares the names of the planes.
   - 4.2. There is no longer a need to pass around the number of the planes in the list, since getCount() will provide it.
   - 4.3. addPlaneInformation() and readPlanes() will take a little reworking, and utilize a local Plane variable.
5. Create an empty Airport class that is publicly derived from City. (5 minutes)
   - 5.1. Change the private data in City to protected.
   - 5.2. Once you have created the class, use a global search and replace to change all mentions of City in main.cpp to Airport.
   - 5.3. Remove the #include "city.h", and add #include "airport.h" to the top of main.cpp.
6. Enhance the Airport class so that it can store the information contained in an airline file whose name is passed as a command line argument. (20 minutes)
   - 6.1. An airline file has the following format:
     - 6.1.1. The first ten lines should be ignored. They contain valid values used for option 6 for the menu, and may be used by you to test your program.
     - 6.1.2. All succeeding lines contain an origin airport followed by the number of flights from the airport, and then airline/destination airport pairs.

6.1.3. There are seven airlines: United, Delta, Alaska, Southwest, JetBlue, SkyWest, and Virgin.

6.2. Add a List <Flight> to the Airport to keep track of the flights of the airlines from it.

6.2.1. Create Flight class in airport.h that has Airport as a friend.

6.2.2. Since Flight will be used in a List, you will need to write an overloaded < for it. You may write any sort for it now, but the next step will want it to reverse the order of the entries—a little puzzle for you.

6.3. Add a readAirlines() function to main.cpp that is called from main(). It is up to you to decide how you will read and insert the flight information into the Airport list. I only read the origin in readAirlines(). The rest was read in an Airport method.

7. Add an option to the menu to print out the airport abbreviations and their flights.
   7.1. Add overloaded << operator friends to the List, Airport, and Flight classes so that run() in main.cpp need only have the line "cout << cities;" in the switch statement for option 6 to be able to print out the entire list.
   7.2. The << operator that takes a List<T> as a parameter is template function, and must be a friend of both List and ListNode classes. The syntax for making a template function a friend is a little tricky. Look at http://stackoverflow.com/questions/18792565/declare-template-friend-function-of-template-class, or page 665 in the text for help.
8. Add an option to the menu to find a route between two airports, that calls determineRoute() of main.cpp. (35 minutes)
   8.1. You may assume that the user will enter valid airports, and airline, though there may not be a route between them for the given airline.
   8.2. Since you will now have multiple instances of List<Airport>, the static count variable will be misleading, and of no use anymore. Make both count and getCount() non-static. Get rid of the declaration of count at the top of list.cpp. Make sure you initialize count to zero in the List constructor. If you have any instances where you call getCount() with a class name, you will need to change them to actually using a List object (I had four).
   8.3. To find the route you will need to call a recursive Airport function called findRoute() that takes, among other parameters, the original List<Airport> cities, and another List<Flight> to hold the route. To preserve the proper order of the route, you will need to play with operator< of the Flight class.

shapes.cpp (25 minutes, 10 points)

I adapted this program from an early ECS 30 assignment. For this assignment you will write a base class, Shape, that has only two functions specified in it: operator<<, and a pure virtual write(). You are to write five classes that are derived from Shape that implement write(): IntersectingLines, BaseLine, ParallelLines, BlankLines, and Circle. Using the classes your program must draw the house and woman of my executable.

Specifications:

1. main() will only contain:
   1.1. A declarations of an array of nine Shape*.
   1.2. Nine assignment of new derived class objects.
   1.3. A for-loop that calls: "cout << shapes[i];" nine times
   1.4. "return 0;"
2. shapes.cpp will contain all of the class declarations. You may provide the implementation of write() within the classes themselves.
3. Note that the write() methods of all the classes need not be public, but Shape must be publicly derived.
4. There is no need for a Makefile for this program.

General programming standards beyond those provided on the Programming Standards handout.

1. main() in main.cpp may only contain a variable declarations, function calls, and a return statement.
2. All functions must be implemented in .cpp files.
3. Use const wherever possible in function parameters.
4. All data must be private or protected. Only overloaded iostream operators, List for ListNode, and Flight for Airport, may be friends.
5. Enclose all of your header file code in #ifndef .. #endif preprocessor blocks.
6. Your Makefile must use g++ with –ansi, -Wall, and –g on all compiling lines.
7. In simulator.out, all dynamically allocated memory must be freed before reaching the return statement in main().

```
[ssdavis@lect1 p6]$ shapes.out
    /\
   /  \
  /    \
 /      \
/        \
----------
|        |
|        |
|        |
----------


    *
  *   *
   * *
    /\
   /  \
  /    \
 /      \
/        \
----------
    /\
   /  \
  /    \
 /      \
/        \
[ssdavis@lect1 p6]$
```

```
[ssdavis@lect1 p6]$ head -11 airlines-1.txt
United COS SCK
Delta SEA FUL
Alaska DET TPA
Southwest SCK LFT
JetBlue SBN GSO
SkyWest BIL SGF
Virgin JAN NEW
United NEW ICT
Delta DSM ANC
Alaska NGU COS
GRB 11 United OJC United PUB Delta BOS Southwest PVU Southwest WJF SkyWest FAT SkyWest
IND SkyWest CRP Virgin FWA Virgin ELP Virgin RAL
[ssdavis@lect1 p6]$
```

```
[ssdavis@lect1 p6]$ simulator.out airlines-1.txt

Flight Simulator Menu
// Edited by Sean
6. Show airline flights.
7. Determine route between two airports.

Your choice (0 - 7): 6
GRB: Vi-RAL Vi-ELP Vi-FWA Sk-CRP Sk-IND Sk-FAT So-WJF So-PVU De-BOS Un-PUB Un-OJC
BIL: Sk-NEW Sk-LRD Sk-PDX Je-AUS De-NGU De-OKC De-STL Un-BOS
BUR: So-MGM Al-HNL Al-MOD Al-NEW
ERI: Je-FNT De-PHX De-PIT
PUB: Je-GRB Je-JAX Je-ICT De-LAN Un-FAT Un-AHN Un-TOA
MAF: Je-FAY Je-MKE Al-PIT De-MEM De-PIE
SBN: Je-BRO So-SFO
OUN: Vi-CAE So-LAX Al-HSV Al-BOS Al-PVU De-CKV De-MKE De-EMT
ABE: Sk-RFD Sk-LFT Sk-MSN Al-SLC Al-RIC De-BAL
// Edited by Sean
 IND: Vi-IAH Vi-AMA Sk-BDR Je-ABQ Je-LRD Je-PIE So-CRP So-FSD So-DCA Un-LRD Un-RNO
SJC: Sk-PIA Sk-PHL Je-STL Al-LEX Al-FNT Un-LOU
DAL: Je-BOI Je-SLE Un-DEN Un-SAT Un-SLC
SDM: Vi-BAL Vi-CID Vi-EVV Sk-AGS Sk-OMA Sk-OAK So-BDR So-SNS So-RDU Un-ATL Un-LAN Un-MSN
SAT: Vi-TPA Vi-ARR So-LBB So-ERI So-HFD Un-WJF Un-PHL Un-CAE
PHL: Je-ARB Je-LAS So-ORH So-FAT So-LAN De-PVU De-SNS
PHX: Sk-BIL Je-ROC Je-ONT Je-RNO So-RNO So-HVN Al-FTW Al-HFD
IAH: Vi-CVG Sk-ABQ Sk-CHI So-EVV Al-PIE Un-CHA
CHI: Sk-PMD Al-SMF Al-JAX De-SFF De-MHT De-DCA Un-CLE Un-OXR Un-SHV
LAX: Sk-CMH So-NGU So-HWD So-BNA Un-OAK Un-ARB
NYC:

Flight Simulator Menu
// Edited by Sean
7. Determine route between two airports.

Your choice (0 - 7): 7

Please enter origin destination and an airline: SCK LFT Southwest
So-SCK So-W39 So-ROC So-CCR So-DET So-LFT

Flight Simulator Menu
// Edited by Sean
7. Determine route between two airports.

Your choice (0 - 7): 7

Please enter origin destination and an airline: DSM ANC Delta
De-DSM De-ORL De-CLT De-LNK De-AUS De-ANC

Flight Simulator Menu
// Edited by Sean
7. Determine route between two airports.

Your choice (0 - 7): 7

Please enter origin destination and an airline: SFO LAX Southwest
No route found.

Flight Simulator Menu
// Edited by Sean
7. Determine route between two airports.

Your choice (0 - 7): 0
[ssdavis@lect1 p6]$
```