# 

# CSE 240: Course Project

*Flight Booking System*

**Group of 3 to 4 members are required to complete this project.**

# Project Objective:

The objective of this project is to engineer an airline flight reservation system using C++. The reservation system should be able to book and purchase flights from list of commercial airlines (e.g. Southwest, US Airway, or Delta.)

# Reservation System Specification:

## Input Load

The system must take 2 input files (Hub.csv and Flight.csv) to populate the flight and hub information before any reservations can be made. These files are a comma delimited files (with .csv extension), which are commonly used in the industry. A detail description of each file is provided below:

### Hub.csv

This file contains the location of all airport terminals (hubs) used by the reservation system. The Hub.csv file contains the following information:

* + Column 1: The name of the airport (e.g. Phoenix International Airport). In this project, airport terminals are also called hubs. (Hubs are unique identifies.)
  + Column 2: The full name of the city (e.g. Phoenix or Chicago). These are called locations.

### Flight.csv

The file contains all the flight information used by the reservation system. The Flight.csv file contains the following information:

* + Column 1: Flight number
  + Column 2: Ticket price. (The ticket price does not include baggage fees.)
  + Column 3: Source hub (airport terminal).
  + Column 4: Destination hub (airport terminal).
  + Column 5: Departure Date and Time (Minutes/Hours / Day /Month/Year) (Hint: You will need to build a special class to capture this information.)
  + Column 6: Flight duration (in minutes). (This does not include flight delays.)

## Route Calculations (Shortest Route vs. Cheapest Route)

To make the implementation easier, all trips will start from Phoenix International Airport. Users will select their destination; their expected date and expected time of departure; number of bags; and filter type (either lowest price or shortest time to destination).

The reservation system will take the user’s input and determine the best route. You will only need to return the top solution. To meet project specification, your search algorithm must search through at least three (3) depth level. (For example, your lowest price ticket from Phoenix to Chicago could be going from Phoenix 🡪 Los Angeles 🡪 Denver 🡪 Chicago.)

### Baggage Fees

When trying to calculate the cheapest route, you must also include baggage fees associated with each company. Formulas are provided below:

* + Southwest: $25 per luggage.
  + US Airway: 1st luggage is free; $25 for each additional luggage.
  + Delta: No luggage fees.

### Flight Delays

When trying to calculate the shortest route, you must also include possible delays during layover. Formulas are provided below, please use the departure time to determine the delay:

* + Southwest: 7:00AM to 5:00PM (30 minutes delay) ; 5:01PM to 6:59AM (no delay)
  + US Airway: 7:00AM to 5:00PM (10 minutes delay); 5:01PM to 1:00AM (15 minutes delay); 1:01AM to 6:59AM (no delay).
  + Delta: 20 minutes delay

## Confirmation (Printing Itinerary)

Upon confirmation, the reservation system will need to print the itinerary, which includes displaying the tickets. Please following the template below:

Flight Number Company Source Location Departure Date/Time

Destination Location Arrival Date/Time (including delays)

Price – Show Calculation Details

Number of Bags

Running Grand Total

Here’s a sample output for reference:

DT1005 Delta Phoenix 3/3/2014 08:00

San Diego 3/3/2014 10:20

$350.00 Base Price = $350.00

UA2004 US Airway San Diego 3/3/2014 12:00

San Francisco 3/3/2014 17:10

$450.00 Base Price + $50 Baggage Fees = $500.00

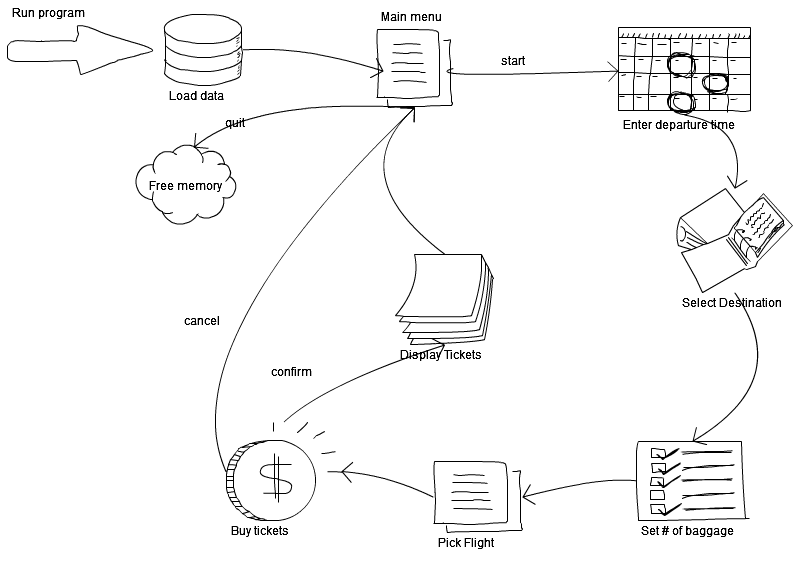
Check-in 3 Bags

Grand Total: $850.00

(The itinerary must be printed in descending order; starting with the source destination, connecting flights, and then finally the destination.)

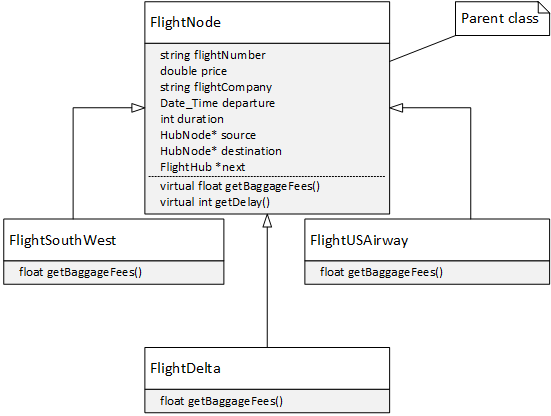
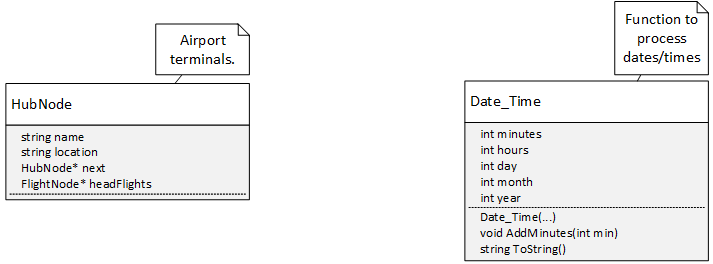
## Reservation Flowchart

A flowchart is provided to help animate the process from searching to booking and confirming your itinerary.

****

## Class Diagram

To help you with this project, we have provided a sample class diagram depicting the relations between different key objects. *\*Note: This is* ***not*** *the complete class diagram. We are only highlight key information that are critical for this project. You may need additional class or class members – including functions.*



HINT: Additional functions might be needed for the derived class…

# Allow n-Hops to Find the Best Flight

To meet the normal project specification, your search algorithm must search through at least three connecting flights. For 50 points extra credit, you are to design and implement a more complicated algorithm that can determine the cheapest/shortest route for a trip with any number of connecting flights (hops). You will need to select some algorithm that solves the ‘shortest path’ problem ((<http://en.wikipedia.org/wiki/Single-source_shortest_path_problem>) and implement it for this particular domain. This means modifying it (or maybe the data structure) to support the departure time constraints. It is likely that you may not be able to find the optimal solution, but, if your algorithm is sufficiently close, it is acceptable.

Be aware that this may be quite challenging should be attempted after the core functionality is in place.

# Programming Specifications:

* C++ must be used to program this project. Visual Studios is highly recommended for this task.
* **Classes** must be usedin the project. ***(struct is forbidden!!!)***
* Demonstrate the use of **polymorphism** and **inheritance** when creating the hubs and the flight company.
* A **linked list** must be used to manage the airlines.
* **Enumeration** and **type definition** must be implemented.
* Proper use of **memory management** must be demonstrated. (For every *new* called there must be a corresponding *delete* to free it from the heap memory.)
* Establish proper error handling and use of exceptions using throw and catch. Such as catching 25 hours in a day or 367 days in a year.

# Project Deliverables

This project has three (3) deliverables. You are required to submit working code for each deliverables.

1. First deliverable is due Saturday, **March 22nd**. For milestone 1, we expect to see the following completed:

* Create a class called Date\_Time. It must include members, for minutes, hours, day, month, and year.
* Create a class called HubNode that implements a linked list node. (Please refer to the class diagram for additional information.)
* Create a class called FlightNode that implements a linked list node. (Please refer to the class diagram for additional information.)
* A debugging function that loops over each hub and prints the outgoing flights.
* All dynamic memory allocations must have the corresponding deallocation.
* Reminder: Program must be built using C++.

1. The second deliverable is due Saturday, **March 29th**. In addition to the content of the first deliverable, the sorting of the time and the cost should be finished.
2. The last deliverable is due Saturday, **April 5th**. It is the working system.