

COP 4814 Assignments

Each team member must upload a separate copy of each assignment, to avoid receiving a grade of zero.

Assignment 1

Creating the flights.xml File

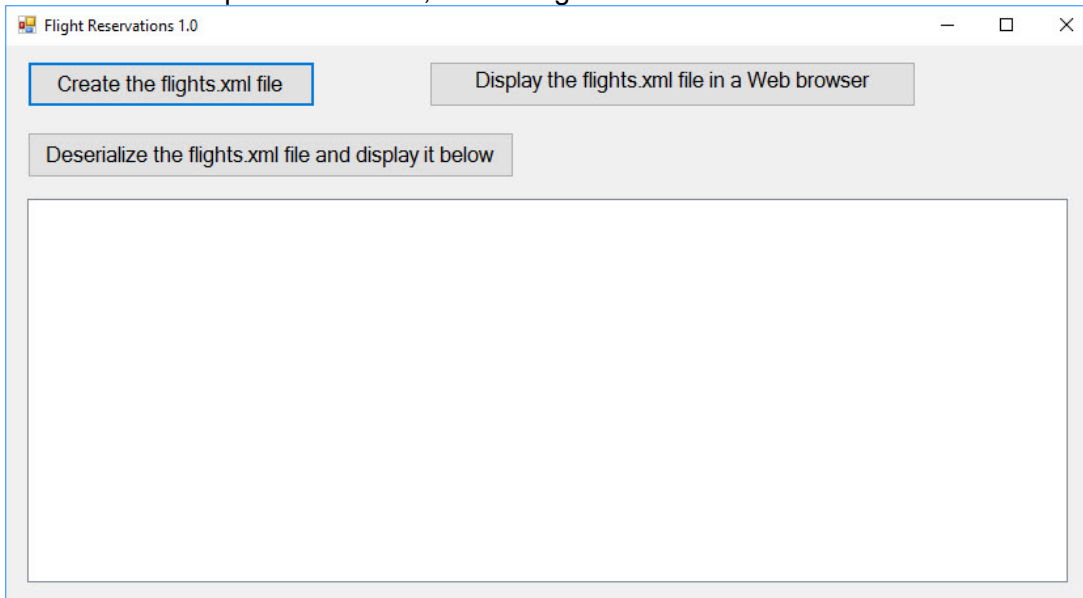
Write a C# program that uses serialization to create your own flights.xml file. Your C# program must create this file so your later assignments can regenerate this file when you test your flight reservations. Watch my Serialization videos to learn how to create XML files. Create a component (class library) to hold all code not directly used to display data on a form. In your component, you will need to create a Flight class and a FlightFactory class (to handle serialization). Link to your component from the GUI program, using the "References" item in the Solution Explorer window.

From the "Flight Reservation System Documents" page in Moodle, download my starting version of the application, which has code to display a short sample xml file in a Web browser window. Here are some details to keep in mind regarding the flights.xml file:


Your flights.xml file must be created in the same folder as your CreateFilesForm. Each aircraft holds 10 passengers. The file must contain four flights in each direction on all the possible routes involving these four airports: MIA, SEA, DEN, and LAX, for dates 6/1/2017 through 7/1/2017, inclusive. You will probably want to use a loop when creating this file, possibly with some random numbers.

The flight number and time schedule must be the same each day. Flights between different cities have different fares. Flights between the same two cities (MIA-DEN, for example) must have different fares at each time of the day. (Some times are more desirable than others.) Fares for flights will never be less than \$500 and never more than \$3,000. Flights in the middle of the day tend to be more expensive than early morning and late evening flights.

Here is the startup user interface, containing 3 buttons and a ListBox:



This is partial sample output in the browser form after clicking the second button on the top:



The screenshot shows a web browser window with a single tab. The address bar displays the file path: file:///D:/Dropbox/Classes/Spr_2017/cop 4814/assignments/Solution - Asg 1/Flight Reservations/Flight Reservations/bin/Debug... The browser content area displays XML data. The XML starts with a declaration: <?xml version="1.0" encoding="utf-8" ?>. It then features a root element <ArrayOfFlight xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">. This root element contains four child elements, each labeled <Flight>. Each <Flight> element contains several sub-elements: <FlightNumber>, <DateAndTime>, <OrigAirport>, <DestAirport>, <Fare>, and <Seats>. The data for these elements varies across the four flights. For example, the first flight has FlightNumber 102, DateAndTime 2017-06-01T06:10:00, OrigAirport MIA, DestAirport SEA, Fare 1750, and Seats 10. The second flight has FlightNumber 104, DateAndTime 2017-06-01T11:30:00, OrigAirport MIA, DestAirport SEA, Fare 2350, and Seats 10. The third flight has FlightNumber 104, DateAndTime 2017-06-01T14:30:00, OrigAirport MIA, DestAirport SEA, Fare 2450, and Seats 10. The fourth flight has FlightNumber 106, DateAndTime 2017-06-01T20:40:00, and its other elements are partially visible. The browser window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.

```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfFlight xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <Flight>
  <FlightNumber>102</FlightNumber>
  <DateAndTime>2017-06-01T06:10:00</DateAndTime>
  <OrigAirport>MIA</OrigAirport>
  <DestAirport>SEA</DestAirport>
  <Fare>1750</Fare>
  <Seats>10</Seats>
</Flight>
- <Flight>
  <FlightNumber>104</FlightNumber>
  <DateAndTime>2017-06-01T11:30:00</DateAndTime>
  <OrigAirport>MIA</OrigAirport>
  <DestAirport>SEA</DestAirport>
  <Fare>2350</Fare>
  <Seats>10</Seats>
</Flight>
- <Flight>
  <FlightNumber>104</FlightNumber>
  <DateAndTime>2017-06-01T14:30:00</DateAndTime>
  <OrigAirport>MIA</OrigAirport>
  <DestAirport>SEA</DestAirport>
  <Fare>2450</Fare>
  <Seats>10</Seats>
</Flight>
- <Flight>
  <FlightNumber>106</FlightNumber>
  <DateAndTime>2017-06-01T20:40:00</DateAndTime>
```

This is partial sample output in the ListBox after clicking the third button

102,	06/01/2017	at 06:06,	MIA	to SEA,	Cost: 1750.0,	Seats: 10
104,	06/01/2017	at 11:06,	MIA	to SEA,	Cost: 2350.0,	Seats: 10
106,	06/01/2017	at 14:06,	MIA	to SEA,	Cost: 2450.0,	Seats: 10
108,	06/01/2017	at 20:06,	MIA	to SEA,	Cost: 2000.0,	Seats: 10
122,	06/01/2017	at 08:06,	SEA	to MIA,	Cost: 2250.0,	Seats: 10
124,	06/01/2017	at 10:06,	SEA	to MIA,	Cost: 2350.0,	Seats: 10
126,	06/01/2017	at 17:06,	SEA	to MIA,	Cost: 2450.0,	Seats: 10
128,	06/01/2017	at 22:06,	SEA	to MIA,	Cost: 1850.0,	Seats: 10
132,	06/01/2017	at 08:06,	SEA	to DEN,	Cost: 1250.0,	Seats: 10
134,	06/01/2017	at 10:06,	SEA	to DEN,	Cost: 1350.0,	Seats: 10
136,	06/01/2017	at 17:06,	SEA	to DEN,	Cost: 1450.0,	Seats: 10
138,	06/01/2017	at 22:06,	SEA	to DEN,	Cost: 850.0,	Seats: 10
142,	06/01/2017	at 06:06,	DEN	to SEA,	Cost: 1200.0,	Seats: 10
144,	06/01/2017	at 12:06,	DEN	to SEA,	Cost: 1450.0,	Seats: 10
146,	06/01/2017	at 15:06,	DEN	to SEA,	Cost: 850.0,	Seats: 10
148,	06/01/2017	at 21:06,	DEN	to SEA,	Cost: 750.0,	Seats: 10
162,	06/01/2017	at 07:06,	MIA	to DEN,	Cost: 1150.0,	Seats: 10
164,	06/01/2017	at 12:06,	MIA	to DEN,	Cost: 1550.0,	Seats: 10
166,	06/01/2017	at 18:06,	MIA	to DEN,	Cost: 950.0,	Seats: 10
168,	06/01/2017	at 22:06,	MIA	to DEN,	Cost: 750.0,	Seats: 10
172,	06/01/2017	at 06:06,	DEN	to MIA,	Cost: 1200.0,	Seats: 10
174,	06/01/2017	at 12:06,	DEN	to MIA,	Cost: 1450.0,	Seats: 10
176,	06/01/2017	at 15:06,	DEN	to MIA,	Cost: 850.0,	Seats: 10
178,	06/01/2017	at 21:06,	DEN	to MIA,	Cost: 750.0,	Seats: 10
182,	06/01/2017	at 06:06,	DEN	to LAX,	Cost: 1200.0,	Seats: 10
184,	06/01/2017	at 12:06,	DEN	to LAX,	Cost: 1450.0,	Seats: 10
186,	06/01/2017	at 15:06,	DEN	to LAX,	Cost: 850.0,	Seats: 10
188,	06/01/2017	at 21:06,	DEN	to LAX,	Cost: 750.0,	Seats: 10
202,	06/01/2017	at 07:06,	LAX	to DEN,	Cost: 750.0,	Seats: 10
204,	06/01/2017	at 12:06,	LAX	to DEN,	Cost: 550.0,	Seats: 10
206,	06/01/2017	at 18:06,	LAX	to DEN,	Cost: 450.0,	Seats: 10
208,	06/01/2017	at 22:06,	LAX	to DEN,	Cost: 350.0,	Seats: 10
212,	06/01/2017	at 07:06,	MIA	to LAX,	Cost: 2150.0,	Seats: 10
214,	06/01/2017	at 12:06,	MIA	to LAX,	Cost: 2525.0,	Seats: 10
216,	06/01/2017	at 18:06,	MIA	to LAX,	Cost: 1950.0,	Seats: 10
218,	06/01/2017	at 22:06,	MIA	to LAX,	Cost: 1750.0,	Seats: 10
222,	06/01/2017	at 06:06,	LAX	to MIA,	Cost: 2200.0,	Seats: 10
224,	06/01/2017	at 12:06,	LAX	to MIA,	Cost: 2450.0,	Seats: 10
226,	06/01/2017	at 15:06,	LAX	to MIA,	Cost: 1825.0,	Seats: 10
228,	06/01/2017	at 21:06,	LAX	to MIA,	Cost: 1750.0,	Seats: 10
232,	06/01/2017	at 07:06,	SEA	to LAX,	Cost: 650.0,	Seats: 10
234,	06/01/2017	at 12:06,	SEA	to LAX,	Cost: 820.0,	Seats: 10
236,	06/01/2017	at 18:06,	SEA	to LAX,	Cost: 750.0,	Seats: 10
238,	06/01/2017	at 22:06,	SEA	to LAX,	Cost: 620.0,	Seats: 10
242,	06/01/2017	at 06:06,	LAX	to SEA,	Cost: 599.0,	Seats: 10
244,	06/01/2017	at 12:06,	LAX	to SEA,	Cost: 815.0,	Seats: 10
246,	06/01/2017	at 15:06,	LAX	to SEA,	Cost: 799.0,	Seats: 10

Use the ListBox's Anchor property to allow the ListBox to expand and contract when the user modifies the form size at runtime (by grabbing the window's corner with the mouse). To get full credit for this assignment, you must use a fixed font (Courier New, 11 pt) in the ListBox, and use custom .NET string formatting to align all the columns. Notice that the fare field uses zero-suppression. [Here's where to read about custom numeric formats](#), and [here is where to read about Custom Date and Time formats](#).

Any other details will be discussed in class. You are responsible for attending class and participating in these discussions.

Assignment 2

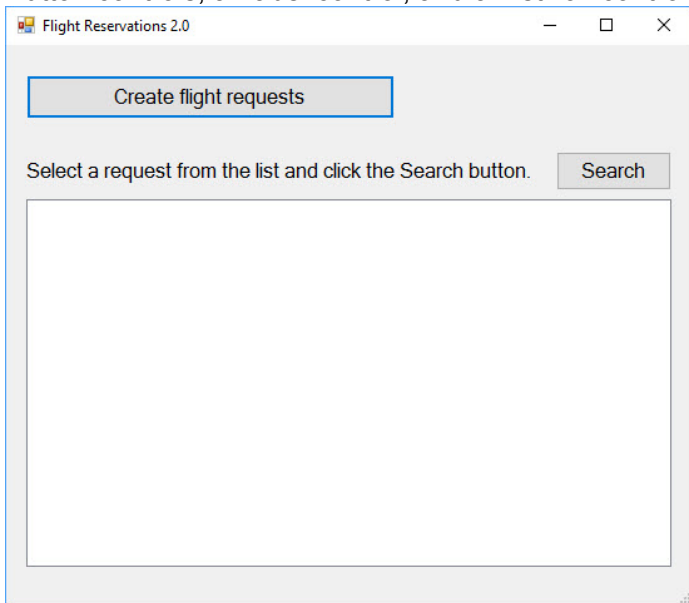
Searching for Flights

In this assignment you continue the program you created for Assignment 1. Your task is to search for flights. First, you need a Main form that starts the program and lets the user open separate windows for the and second assignments. Your form should look as much as possible like the one shown here. The hyperlinks are displayed by LinkLabel controls, by the way. Notice the heading, which references

version 2.0:

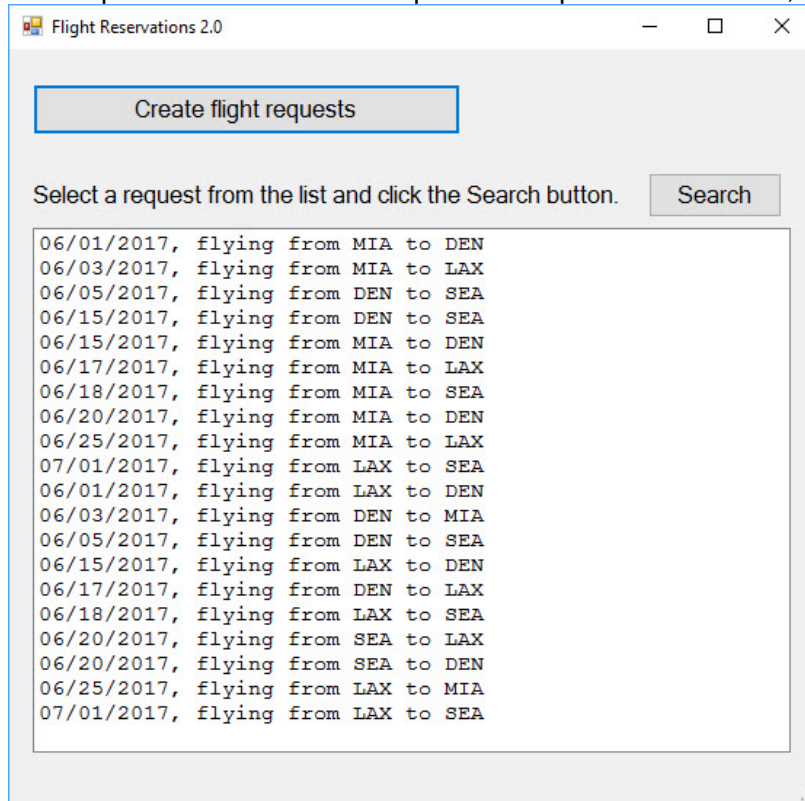


When they click the second link, they are taken to a separate form for Assignment 2, containing two Button controls, a Label control, and a ListBox control:



When the user wants to fly between two cities on a certain date, we call this a "flight request". The *Create flight requests* button shown here will create 20 flight requests and place them in the list box.

The requests should include all possible airport combinations, on a variety of dates. Here is a sample:



Next, the user selects one of the listbox items and clicks the *Search* button. The program displays a Web browser window (the same one from Assignment 1) containing a list of matching flights, in XML format. For example, the user selected the requested trip from MIA to DEN on June 15:



Notice that only matching flights are displayed, not the entire flights.xml file. Your code that implements the search must create a new xml file containing only the matching flights. Give it some new name, such as matchingflights.xml. That will be the file you display in the browser form.

Notes:

Use a Form_Load event handler on your windows forms to initialize variables. You create a handler by double-clicking any open area of the form in design mode.

In a ListBox control, the SelectedIndex property tells you which row was selected by the user.

I would suggest creating the following classes for this assignment:

Shared: contains the list of flights, which tends to be shared between different component classes.

This class must not be Public, because we do not want it to be accessed by any code outside the Reservation Library component.

FlightRequest: contains a date and two airport codes, used when searching for matching flights. This class must be Public.

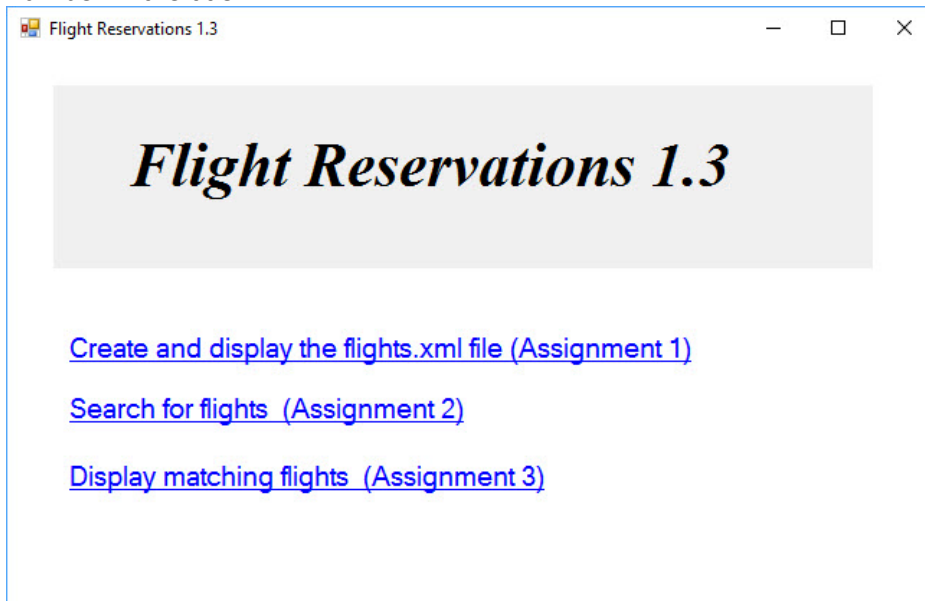
ReservationFactory: all methods related to searching for flights, including the creation of the FlightRequest objects. This class will expand in future assignments, as we begin to schedule reservations. This class must be Public, and it does not have the same structure as the Flight class.

Assignment 3

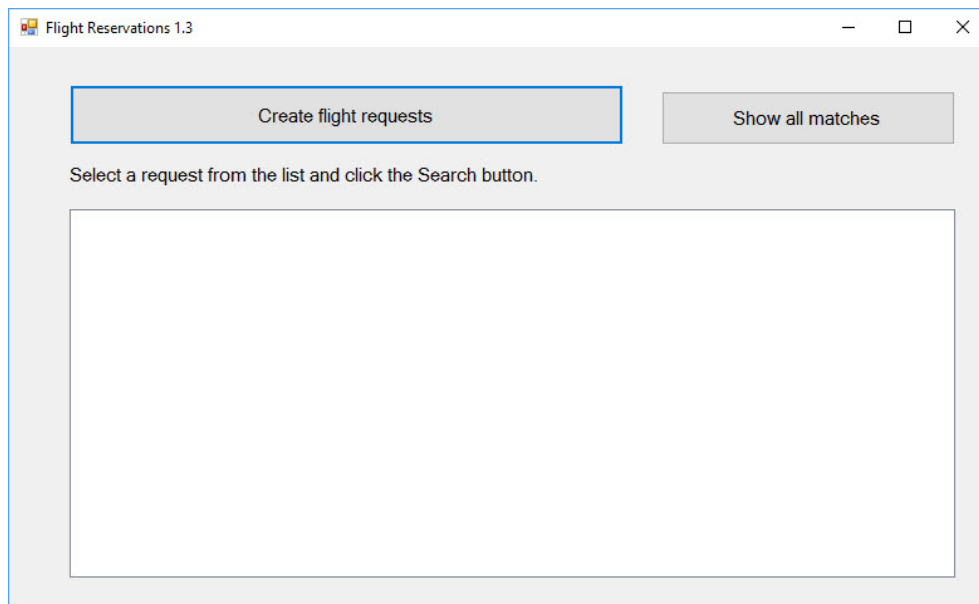
Displaying Matching Flights in HTML

In this continuation of the Flight Scheduling program, you will display all matching flights in a nice looking HTML table. The HTML is created as a result of matching the XML file you created in Assignment 2 against an XSLT file.

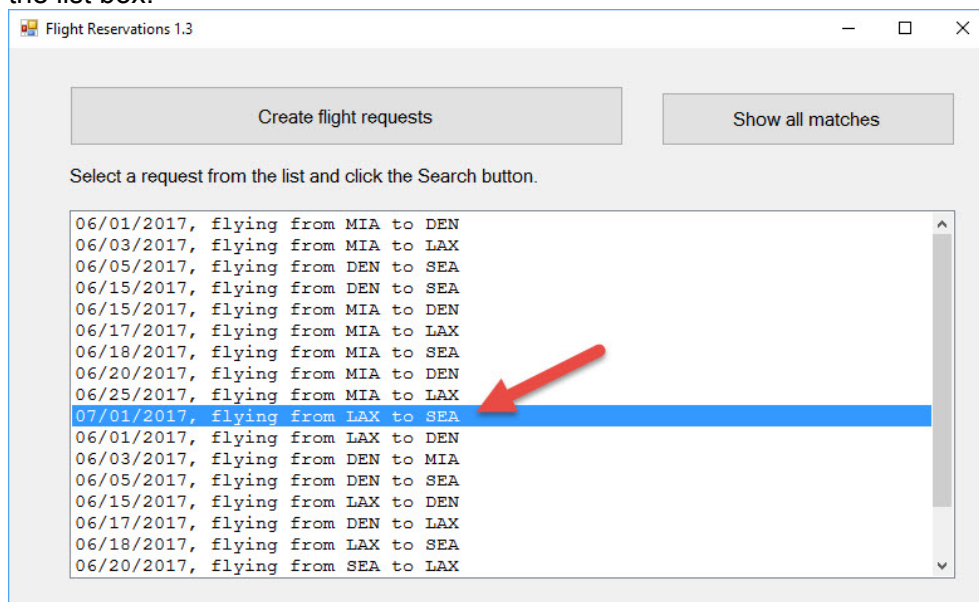
Start by adding a new link on the program's main form and change the program's version number in the title:



This will take you to a slightly modified version of the Search for Flights form you created for Assignment 2:



As before, the user clicks the first button, views a list of flight requests, and selects one from the list box:



When the user clicks the Show all matches, button, the program merges the xml file (created in Assignment 2) with a XSLT file, and produces an HTML file. It then displays this file in the popup browser window:

Matching Flights: LAX to SEA

Flight Number	Date/Time	Route	Fare	Seats
242	7/1/2017, 06:40:00	LAX to SEA	599.00	10
248	7/1/2017, 21:10:00	LAX to SEA	705.00	10
246	7/1/2017, 15:05:00	LAX to SEA	799.00	10
244	7/1/2017, 12:30:00	LAX to SEA	815.00	10

Notice how the desired route appears in the H2 heading at the top. Also, notice that the prices are sorted in ascending order, and they are formatted to two decimal places. All columns must be centered, and the colors and other visual styles must be as shown.

Here's an example where the user selects a different flight request from the list box:

Matching Flights: DEN to SEA

Flight Number	Date/Time	Route	Fare	Seats
142	6/15/2017, 06:40:00	DEN to SEA	1,200.00	10
144	6/15/2017, 12:30:00	DEN to SEA	1,450.00	10
148	6/15/2017, 21:10:00	DEN to SEA	750.00	10
146	6/15/2017, 15:05:00	DEN to SEA	850.00	10

Assignment 4

Multi-City Flights

In this version of the application, use the same list of flight requests you created for Assignments 2 and 3. When the user selects a request from the List box, they will see list of multi-city flights that meet their flight request requirements. (In my terminology, a *multi-city flight* is a combination of two connecting flights.) Here are some constraints to make your life easier:

1. Every flight lasts 4 hours.
2. You can only choose a connecting flight that leaves 1 hour or more after the arrival of the first flight.
3. A connecting flight must leave on the same date as the originating flight.
4. All airports are in the same time zone.
5. All flight times and flight durations are to be displayed in 24-hour format, as 05:40 or 14:20.

For example, suppose the user selects a flight request to travel from MIA to SEA. The following multi-city flights can be displayed because at least 5 hours pass between the departure time of the first flight and the departure time of the second flight:

```
MIA-LAX (09:00), followed by LAX-SEA (14:00)
MIA-LAX (09:00), followed by LAX-SEA (18:00)
MIA-LAX (13:00), followed by LAX-SEA (18:00)
MIA-LAX (13:00), followed by LAX-SEA (20:00)
MIA-LAX (15:00), followed by LAX-SEA (20:00)
MIA-DEN (10:00), followed by DEN-SEA (15:00)
MIA-DEN (10:00), followed by DEN-SEA (19:00)
MIA-DEN (14:00), followed by DEN-SEA (19:00)
MIA-DEN (14:00), followed by DEN-SEA (22:00)
MIA-DEN (17:00), followed by DEN-SEA (22:00)
```

But the following multi-city trips would not be displayed because the passenger would not be able to make the connection:

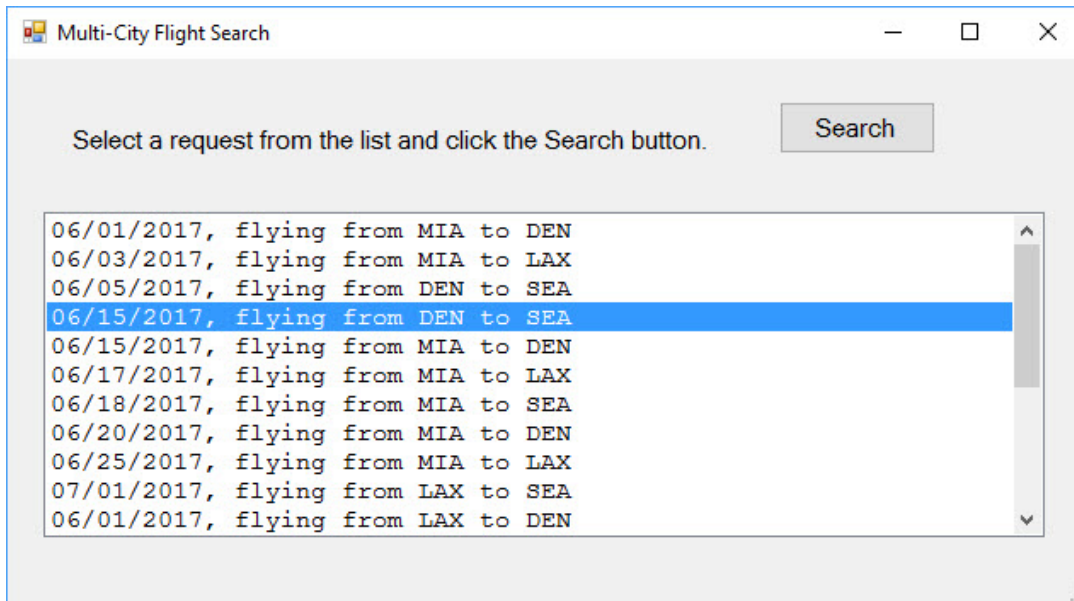
```
MIA-LAX (13:00), followed by LAX-SEA (14:00)
MIA-LAX (15:00), followed by LAX-SEA (18:00)
MIA-DEN (14:00), followed by DEN-SEA (15:00)
MIA-DEN (17:00), followed by DEN-SEA (19:00)
```

In a list box, you must display detailed information about each trip: For each flight, display the flight number, airport codes, and departure time. After the two flights, display the trip's total duration and cost.

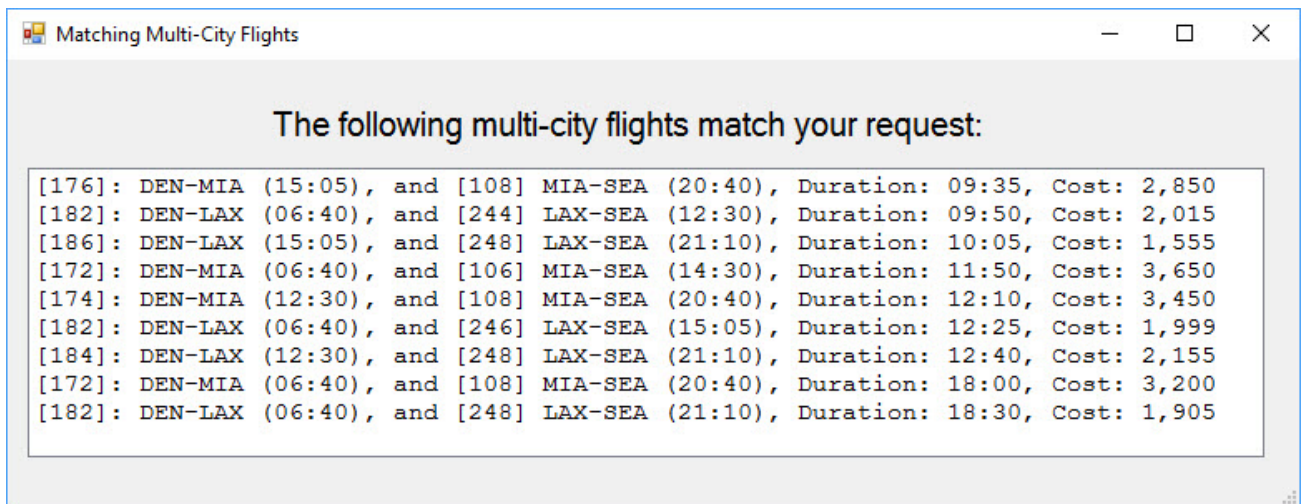
Let's look at a simulated run. The startup form has a new link, and its version number is now 1.4:



The next form displays the same flight requests you used in the previous assignments, and the user will select one of the requests in order to perform a search. Be sure your form's title bar matches the one shown here:

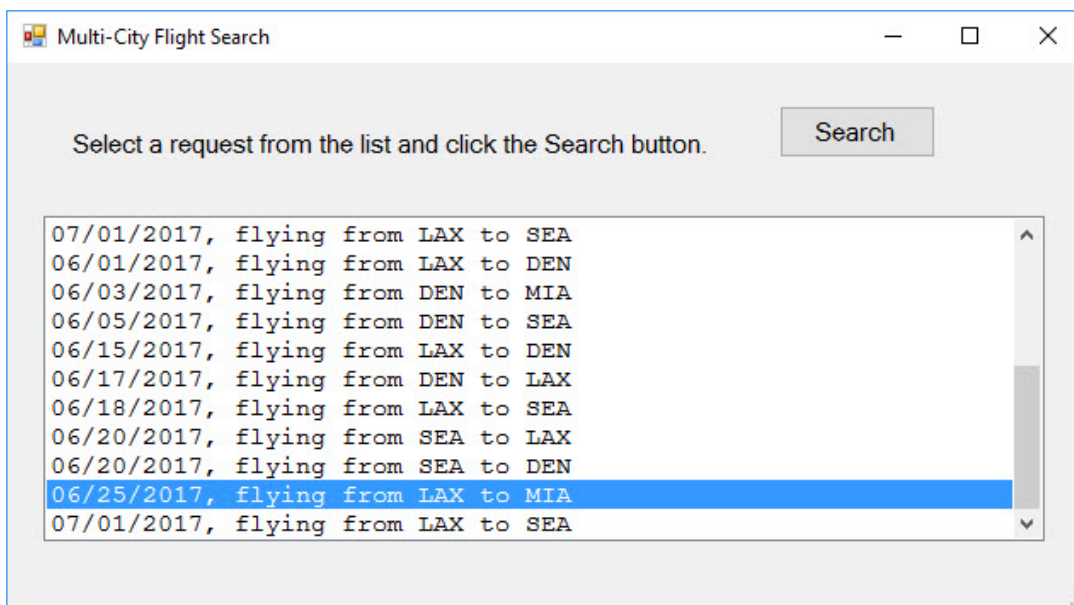


A new dialog window lists all matching multi-city flights, using the requirements explained earlier. The flight ID numbers are enclosed in square brackets [...]. Notice that connecting flights are always 5 hours or more later than the originating flight. Also, the flights are sorted in ascending order by total duration (the combined duration of both flights, plus the layover time). Be sure your form's title bar matches the one shown here:



(To avoid losing 1 point on this assignment, your output must be formatted exactly like mine.)

Now the user can go back and select a different flight request:



And a new list of possible multi-city flights displays in the window:

