COMPRO2 Machine Project
Term 2, AY 2017–2018

Phase 1 Due: March 19, 2018
Phase 2 Due: April 13, 2018

College of Computer Studies

# Flight Booking System *v1.0*

**Flight Booking System** or **FBS** is a system that allows the user to create multi-city tickets. The system has with two modules: Data Module and Booking Module. The Data Module is for the administrator of the system. This is the module where the list of cities, and the list of available flights are managed. The Booking Module are for travellers. The is where the multi-city destination tickets are formed. For FBS (*v1.0*), simply provide the two modules without differentiating the users.

## 1 Modules

### 1.1 Data Module

The Data Module, or DM allows the user to add, delete, view and search cities and flight details.

- DM maintains a list with at most 10 cities. Each city has a name (at least 3 characters to a maximum of 15 characters) and a unique three-letter code.
- DM also maintains a list of 50 available flights. This list contains details such as

  **flight code** a four-character string, where the first two characters are alphabet letters and the last two characters are numerical digits, i.e. XX99. The flight code is unique.

  **source city** a three-letter code. The source city is where the flight will take-off. **destination city** a three-letter code. The destination city is where the flight will land. **departure time** a military time[1] value representing the time the flight will leave the source city. **duration** an integer value that represents the flight duration in minutes. **eta** estimated arrival time in military time format. The value is computed by the system.

- In DM, the user can **add a new city** The user has to enter a unique three-letter code, and the name of the city.

  **delete an existing city** The user chooses a city to be deleted from the list (displayed). Note that all flights with this city will also have to be deleted. **view the list of cities** The user can select to view cities having the specified initial or view a sorted list of cities, sorted alphabetical by city code.

  **search a city** The user can check whether a city code is in the list or not. If the city code is in the list, the city code together with the name of the city will be displayed.

- In DM, the user can also

  **add a flight** The user will enter the flight details. Only flights with unique flight codes can be added into the list. **delete a flight** The user can select from a list the flight to be deleted.

  **view the list of flights** The user can choose to view details of all the flights either arranged alphabetically by flight code, arranged chronological by departure time or arrival time, or arranged in increasing order of duration. Provide navigations, like Previous or Next to allow the user to navigate the display. **search** The user can choose to view flight details by specifying the flight code, the departure city code, or the departure time. Details of the flights will be displayed. Provide navigations, like Previous or Next to allow the user to navigate the display.



[1] Military time is from 0000–2359

- Upon starting the system, the user may opt to load a text file for an initial list of cities or initial list of flights. Note that your system be able to load files from other FBS and vice versa.

| **City File Format** | **Flight File Format** |
|---|---|
| count<br>$code_1$ ␣ $code_1$<br><br>$code_2$ ␣ $code_2$<br>.<br>.<br>.<br>$code_{count}$ ␣ $code_{count}$ | *count*<br>FCODE=*code*$_1$<br>SRCCITY=*source city*$_1$<br>DESTCITY=*destination city*$_1$<br>DEPARTURE=*departure time*$_1$<br>DURATION=*duration*$_1$<br>FCODE=*code*$_2$<br>SRCCITY=*source city*$_2$<br>DESTCITY=*destination city*$_2$<br>DEPARTURE=*departure time*$_2$<br>DURATION=*duration*$_2$<br>...<br>FCODE=*code*$_{count}$<br>SRCCITY=*source city*$_{count}$<br>DESTCITY=*destination city*$_{count}$<br>DEPARTURE=*departure time*$_{count}$<br>DURATION=*duration*$_{count}$ |

- The user can save the list of cities and the list of flights.

## 1.2 Booking Module

The Booking Module (or BM) allows the user to reserve multi-city flights, and view ticket.

- The user can book multi-city flights as long as
  - the destination of the previous flight is the same as the source of the current flight;
  - the eta of the previous flight is at least 15 minutes before the departure time of the current flight;

  • When booking a flight, the user enters the source city and selects from the available flight from that city.
- The user can book a multi-city fight with unlimited number of valid connecting flights.
- While booking of flights is ongoing, the updated multi-city ticket is shown on the screen.
- The user may delete or erase the most recent flight from the current multi-city ticket.
- Booking is complete when the user enters XXX for the source city code.
- When booking is complete, the user has the option to save into a text file the updated multi-city ticket. Each line after the header lists the details of a flight. Each flight details must be center-aligned (biased towards the right) under its corresponding column.

**Multi-City Ticket File Format**

CODE ␣ ETDEPART ␣ SOURCE ␣ DESTIN ␣ ETARRIVAL ␣ DURATION *flight data*$_1$ *flight data*$_2$

...

*flight data*$_n$

# 2 Requirements

1. The City List and Flight Lists in DM must be represented using an array. Use the #define macro for the sizes of the lists.
2. The Multi-City Ticket must be represented as a **linked list of Flights**.
3. The FBS is a **menu-driven system**. The program will only end when the user chooses to exit. Continue? (Yes or No) interface, or anything similar, must not be used for navigation in FBS. (e.g. to add another city in the multi-city ticket, the user will simply enter the City Code. When the user enters XXX, then the multi-city ticket is complete.)
4. Make sure to maintain consistency of data. (e.g. The list of flights must only contain flight to and fromcities that are in your city list.)
5. Make sure that you have implemented the necessary error-checking and verification.
6. Make sure that your implementation has considerable and proper use of arrays/dynamic lists, structures,and user-defined functions. All dynamically allocated spaces must be deallocated properly before program termination.
7. For the bonus: A maximum of 20 points may be given for features over & above the requirements (such asuse of modules in programming or other features not conflicting with the given requirements or changing the requirements). Required features must be completed first before bonus features are credited. Note that use of conio.h, or other advanced C commands/statements may not necessarily merit bonuses.
8. Both **load and save** options in the DM must function properly for this feature to be credited.

# 3 Important Notes

1. Use Dev-C++ to compile your C program. Make sure you test your program completely (compiling & running) in G302 labs.
2. Only programs with **VIEW** working properly will be checked. If the **view** features are not working properly, the project grade will be 0.
3. Do not use brute force. Use appropriate conditional statements properly. Use, wherever appropriate, appropriate loops & functions properly.
4. You may use topics outside the scope of COMPRO2 but this will be self-study. Use of goto label, exit, break (except in switch), global variables are not allowed.
5. Include internal documentation (comments) in your program. On top of each user-defined function, includea description of the function following this format:

```
/*****
    Function:
    Description:
    In-Paramters:
    Returns:
*****/
```

Example[2]:

```
/***** public char charAt(int index) Description:
        Returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char
        value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.
    In Parameters:
```

[2] Taken from http://docs.oracle.com/javase/7/docs/api/java/lang/String.html

*index - the index of the char value.*

*Returns:*

*the char value at the specified index of this string. The first char value is at index 0.*

******/*

6. Bring a printout copy of your source code during your scheduled demo.
   Note that your Source Code must include the internal documentation. The first page of the source code printout should have the following declaration (in comment), the printout of which should be signed:

   */******

   *This is to certify that this project is my own work, based on my personal efforts in studying and applying the concepts learned. I have constructed the functions and their respective algorithms and corresponding code by myself. The program was run, tested, and debugged by my own efforts.*

   *I further certify that I have not copied in part or whole or otherwise plagiarized the work of other students and/or persons.*

   *<your signature>*
   *<your full name>, DLSU ID No.: <number>*

   ******/*

7. Submit your MP **on or before April 13, 2018 (Fri)** in canvas.

8. Use <surnameFirstInit>.c and <surnameFirstInit>.doc as your filenames.

9. Unable to show up on time during the demo schedules, or unable to answer convincingly the questionsduring the demos, will merit a grade of 0.0 for the course.

10. Any requirement not fully implemented or instruction not followed will merit deductions.

11. This project is by pair. Any form of cheating (asking other people's help, copying any part of other's work, etc.) will merit a grade of 0.0 for the course.

12. You may choose your partner but do note that both of you are expected to do the work. If there are any issues between pairs, please let me know ASAP.