# ENEE 150: Intermediate Programming Concepts for Engineers

**Spring 2019** **Handout #18**

## Project #2: ENEE Airlines Flight Database: Due April 1 at 11:59pm

In this project, you will build a C program for the ENEE Airlines flight database. Your program will parse a set of database files that specify the different airports, routes, and flights serviced by ENEE Airlines. After parsing the database files, your program will allow the user to look up information about flights and airports, as well as find different flights that fly between serviced airports.

## 1 Running Your Program

The ENEE Airlines flight database consists of 3 files: airports.txt, routes.txt, and flights.txt. At the beginning of your program, you should try to open the 3 files. If any of the database files cannot be opened, your program should print the error message "Could not open database files," and exit. Otherwise, your program should parse the database files, print a menu of options, and prompt the user for an option. Here is an example of what happens when the program is run (your output should match identically this, and any other output, given in this handout):

```
z:  pr2
Welcome to the ENEE Airlines Flight Database
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:
```

If the user enters a number between 1–4, your program should perform the specified operation, and then prompt the user for another option. This should continue until the user enters the exit option (#5). Your program should validate that every entered option is valid (*i.e.*, a number between 1–5); if not, it should print the error message "Invalid option. Try again." and prompt for another option:

```
z:  pr2
Welcome to the ENEE Airlines Flight Database
Choose an option:
```

```
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  6
Invalid option.  Try again.
Enter option:  -1
Invalid option.  Try again.
Enter option:
```

## 2 Airline Database Files

The 3 airline database files for this project can be downloaded from the course web site (follow the hyperlink labeled "Project 2 files"). The first file, "airports.txt," lists all the airports that are serviced by ENEE Airlines. Each line in this airports file specifies a single airport beginning with the 3-letter airport code, followed by 2 white-space characters, and ending with a string that specifies the airport's location. The airport location string terminates right before the '\n' character at the end of the line. (There may be multiple white-space characters as well as a comma inside this string).

The second file, "routes.txt," gives all the route information for ENEE Airlines. In particular, each line in this routes file lists a pair of airports that are serviced by at least one non-stop ENEE Airlines flight. A line begins with a unique route ID (which is immediately followed by a '.' and 2 white-space characters), and ends with a list of two airports each specified by their 3-letter airport codes (the airport codes are separated by a single white-space character). The first airport code specifies the departure airport while the second airport code specifies the arrival airport for all non-stop flights servicing the associated route ID. The lines in the routes file are sorted by the route IDs. This also orders all routes alphabetically by the departure airport location and then alphabetically by the arrival airport location.

The last file, "flights.txt," lists all the flights maintained by ENEE Airlines. Each line in the file provides the information associated with a single flight. A line begins with the flight number (you may assume each flight has a unique flight number). Next, there is a route ID that specifies the pair of airports from the routes file that the flight services. Then, there is the departure and arrival times for the flight. Each time specifier is in 12-hour format, with a single trailing character, 'a' or 'p', denoting a.m. or p.m. Finally, there is a string that specifies the flight's frequency. If this string is "Daily", then the associated flight runs 7 days a week. If this string starts with a number, then a list of 1 or more numbers between 1–7 are given that specify the days of the week on which the flight runs, with '1' denoting Sunday, '2' denoting Monday, '3' denoting Tuesday, and so on. In this case, the flight does not run on those days missing from the frequency string. And if this string starts with the character 'X', then a list of 1 or more numbers between 1–7 are given immediately following the 'X' character that specify the days of the week

on which the flight *does not* run (using the same association between numbers and days of the week). In this case, the flight runs on those days missing from the frequency string. The lines in the flights file are sorted by the flight number. (Notice, flight numbers are mostly sequential, but there are some flight numbers that are skipped).

To parse the 3 airline flight database files, you will need to declare arrays that will receive all the data. For the purposes of determining array sizes, you may assume there will never be more than 100 airports in the "airports.txt" file, 500 route IDs in the "routes.txt" file, and 3000 flights in the "flights.txt" file.

## 3 Printing Flights

Whenever the user enters '1' at the options prompt, your program should prompt the user for a flight number. If the flight number does not correspond to an existing ENEE Airlines flight, print "Flight <flight number> doesn't exist" and return to the menu. Otherwise, print the information associated with the selected flight. First, print the flight number followed in parenthesis by a list of days of the week on which the flight runs. In the list, use "S" for Sunday, "M" for Monday, "T" for Tuesday, "W" for Wednesday, "Th" for Thursday, "F" for Friday, and "Sa" for Saturday. Then, on separate lines, print the departure and arrival information for the flight. Begin by printing the departure or arrival time for the flight, using the format hh:mm (*i.e.*, both hour and minute are specified using 2 characters) terminated by either "a.m." or "p.m." Then, print the location of the departure or arrival airport followed by the 3-letter aiport code in parenthesis. Here's an example of how your program should respond to option '1':

```
Welcome to the ENEE Airlines Flight Database
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  1
Enter a flight number:  102

Flight 102 (SWThFSa)
    07:05a.m. Orlando, FL (MCO)
    09:31a.m. New York, NY (JFK)

Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
```

```
5.  Exit
Enter option:  1
Enter a flight number:  10000
Flight 10000 doesn't exist
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:
```

# 4 Printing Airports

Whenever the user enters '2' at the options prompt, your program should prompt the user for an airport. You may assume that the user will respond by typing a 3-letter airport code. Your program should first check to see if the airport is serviced by ENEE Airlines. If not, you should print "Airport <airport code> doesn't exist" and return to the menu. Otherwise, your program should print the airport location followed by the airport code in parenthesis. Then, your program should identify all the routes from the routes.txt file for which the airport is either a departure airport or an arrival airport. Let us refer to these as "departure routes" and "arrival routes," respectively. Your program should print "Departures to:" followed by a list of the arrival airports for all of the departure routes. Then, your program should print "Arrivals from:" followed by a list of the departure airports for all of the arrival routes. When printing the airports, you should print the airport location followed by the 3-letter airport code in parenthesis. In each list, the airports should be printed alphabetically by location. (Note, if you print the departure routes and arrival routes in the order you find them in the routes.txt file, they will appear alphabetically by location). Here's an example of how your program should respond to option '2':

```
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  2
Enter airport code:  WWW
Airport WWW doesn't exist
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
```

```
4.  Find 1-stop flights
5.  Exit
Enter option:  2
Enter airport code:  SYR

Syracuse, NY (SYR)

Departures to:
   Fort Lauderdale, FL (FLL)
   New York, NY (JFK)
   Orlando, FL (MCO)

Arrivals from:
   Fort Lauderdale, FL (FLL)
   New York, NY (JFK)
   Orlando, FL (MCO)

Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:
```

## 5 Non-Stop Flights

Whenever the user enters '3' at the options prompt, your program should find and print all of the flights that allow the user to fly non-stop for a particular route and on a particular day. Your program should first prompt the user for the desired route. You may assume that the user will respond by typing two 3-letter airport codes that specify the departure and arrival airports for the route. Your program should first check if either airport entered is not an airport serviced by ENEE Airlines. If so, you should print "Either airport <first airport code> or <second airport code> doesn't exist" and return to the menu. Otherwise, you should prompt the user for a day of the week to travel. You may assume the user will respond with a number between 0–7: 0 denotes travel on *any* day of the week, while 1–7 denotes travel on a specific day of the week.

Your program should identify all non-stop flights on the selected route that run on the specified day(s). If there are no non-stop flights at all, or no non-stop flights that run on the specified day(s), your program should print "Route <first airport code> to <second airport code> has no non-stop flights on the specified days" and return to the menu. Otherwise, your program should print "Non-stop flights from <first airport> to <second airport>:" where the first and second airports consist of the airport location and the

3-letter airport code (in parenthesis). Then, your program should print all the non-stop flights found using the same format for printing flights described in Section 3. If there is more than one non-stop flight to print, you should print the flights ordered by flight number, and separate each flight by an empty line. Here's an example of how your program should respond to option '2':

```
Welcome to the ENEE Airlines Flight Database
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  3
Enter departure and arrival airport codes:  QQQ JFK
Either airport QQQ or JFK doesn't exist
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  3
Enter departure and arrival airport codes:  LGB CUN
Enter a day of the week (1-7, or 0 for all days):  0
Route LGB to CUN has no non-stop flights on the specified days
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  3
Enter departure and arrival airport codes:  BOS LGB
Enter a day of the week (1-7, or 0 for all days):  0

Non-stop flights from Boston, MA (BOS) to Long Beach, CA (LGB):

Flight 481 (SMTWThSa)
    08:00a.m. Boston, MA (BOS)
    11:23a.m. Long Beach, CA (LGB)

Flight 490 (SMTWThFSa)
    05:00p.m. Boston, MA (BOS)
    08:30p.m. Long Beach, CA (LGB)
```

```
Flight 491 (SMTWThFSa)
    05:05p.m. Boston, MA (BOS)
    08:35p.m. Long Beach, CA (LGB)

Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  3
Enter departure and arrival airport codes:  BOS LGB
Enter a day of the week (1-7, or 0 for all days):  6

Non-stop flights from Boston, MA (BOS) to Long Beach, CA (LGB):

Flight 490 (SMTWThFSa)
    05:00p.m. Boston, MA (BOS)
    08:30p.m. Long Beach, CA (LGB)

Flight 491 (SMTWThFSa)
    05:05p.m. Boston, MA (BOS)
    08:35p.m. Long Beach, CA (LGB)

Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:
```

## 6 1-Stop Flights

Whenever the user enters '4' at the options prompt, your program should find and print all of the flights that allow the user to fly with one stop for a particular route and on a particular day. Your program should prompt for the same information (route and day of week to travel) and handle the same error condition (non-existant airports) as described in Section 5. Assuming no errors, your program should identify all pairs of flights that route the user between the departure and arrival airports with exactly 1 stop (*i.e.*, a one-connection flight). In particular, the first flight must originate from the departure airport and terminate at a connecting airport, while the second flight must originate from the connecting airport and terminate at the arrival airport. The connecting airport can be *any* airport serviced by ENEE Airlines. If the user specifies a particular day of the

week to travel (options 1–7), both flights in the pair must run on the specified day; if the user specifies any day to travel (option 0), there must be at least 1 day in common on which both flights run. You must also enforce a minimum and maximum lay-over at the connecting airport: the lay-over must not be less than 1 hour or exceed 2 hours. Lastly, you should only consider lay-overs that span a single day (*i.e.*, the first flight's arrival time and the second flight's departure time should not be on opposite sides of midnight).

If there are no 1-stop flights that meet all of the criteria mentioned above, your program should print "Route <first airport code> to <second airport code> has no 1-stop flights on the specified days" and return to the menu. Otherwise, your program should print "1-stop flights from <first airport> to <second airport>:" where the first and second airports consist of the airport location and the 3-letter airport code (in parenthesis). Then, your program should print all of the 1-stop flights that satisfy the above criteria. Pairs of flights should be printed back-to-back, each using the same format for printing flights described in Section 3. If there is more than one 1-stop flight to print, you should print the flights ordered by the first flight's flight number, and separate each flight pair by an empty line. (If two flight pairs have the same first flight, then order them by the second flight's flight number). Here's an example of how your program should respond to option '4':

```
Welcome to the ENEE Airlines Flight Database
Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:  4
Enter departure and arrival airport codes:  BOS CUN
Enter a day of the week (1-7, or 0 for all days):  0

1-stop flights from Boston, MA (BOS) to Cancun, Mexico (CUN):

Flight 1003 (SMTWThFSa)
    08:00a.m. Boston, MA (BOS)
    09:15a.m. New York, NY (JFK)
Flight 753 (SMTWThFSa)
    10:30a.m. New York, NY (JFK)
    01:45p.m. Cancun, Mexico (CUN)

Flight 1003 (SMTWThFSa)
    08:00a.m. Boston, MA (BOS)
    09:15a.m. New York, NY (JFK)
Flight 754 (SWThFSa)
    10:30a.m. New York, NY (JFK)
    01:45p.m. Cancun, Mexico (CUN)
```

```
Flight 1004 (SMTWThFSa)
    08:15a.m. Boston, MA (BOS)
    09:29a.m. New York, NY (JFK)
Flight 753 (SMTWThFSa)
    10:30a.m. New York, NY (JFK)
    01:45p.m. Cancun, Mexico (CUN)

Flight 1004 (SMTWThFSa)
    08:15a.m. Boston, MA (BOS)
    09:29a.m. New York, NY (JFK)
Flight 754 (SWThFSa)
    10:30a.m. New York, NY (JFK)
    01:45p.m. Cancun, Mexico (CUN)

Choose an option:
1.  Print a flight
2.  Print an airport
3.  Find non-stop flights
4.  Find 1-stop flights
5.  Exit
Enter option:
```

# 7 Termination

Whenever the user enters '5' at the options prompt, your program should terminate.