Project - 2



The project is to load data through a CSV file and write program that will conduct a variety of queries on that data.

# Dataset

The dataset (attached) is a modified CSV file on all International flight departing from US Airports between January and June 2020 reported by the US Department of Transportation (<https://data.transportation.gov/Aviation/International_Report_Passengers/xgub-n9bw>). Each record holds a route (origin to destination) operated by an airline. This CSV file was modified to keep it simple and relatively smaller. Here is a description of each column:

* Column 1 – Month (1 – January, 2 – February, 3 – March, 4 – April, 5 – May, 6 – June)
* Column 2 – 3-letter IATA Airport Code for the origin airport (e.g., SAT for San Antonio International Airport)
* Column 3 – 3-letter IATA Airport Code for the destination airport
* Column 4 – 2-letter IATA Airline Code for the airline (e.g., AA for American Airlines).
* Column 5 – The passenger category, in our example, there is only one category.
* Column 6 – Total number of passengers in that month for that route

Note that there is a header row you must skip. Since this data holds passenger statistics for each route operated by an airline for six months, you should see the airline route repeated six times. For example, you will see the JFK to LHR operated by BA route 6 times, once for each of the six months.

# Task 1 – create route-records.h

All data is loaded into an array of **RouteRecord**’s which will they be queried in **main()**.

* Create a **struct** named **RouteRecord** that will hold information about a route that is operated by one airline. The struct will have the following data members:
  + Origin airport code
  + Destination airport code
  + Airline code
  + Array of passenger counts. There are six months’ worth of data for each route. (Index 0 will represent January’s passenger count, Index 1 will represent February’s passenger count, etc.).
* Add the header guards and prototypes for the functions (see Task 2)
* Include this **enum** in your header file so you can use as values for determining what type of search you will conduct.  
  **typedef enum SearchType { ROUTE, ORIGIN, DESTINATION, AIRLINE } SearchType;**

# Task 2 – create route-records.c

Write the following functions:

* **RouteRecord\* createRecords( FILE\* fileIn )** – This function creates the array of **RouteRecord**’s and initializes it. The function takes in a file pointer. The function will do the following:
  + This function goes through the CSV file and counts the number of total records (not including the header)
  + Dynamically allocate memory for an array of **RouteRecord**’s based on the count.
  + Each **RouteRecord** struct object has an array of 6 integers to hold the number of passengers for six months. Initialize each of these integer values to 0. You do not need to initialize the other data members in the struct.
  + **Rewind** the file pointer
  + Return the pointer to the array you dynamically allocated.
* **int fillRecords( RouteRecord\* r, FILE\* fileIn )** – This function will process the data in the CSV file. Essentially, the code will go through each record, parse out the record, and enter it into the array. The function will follow these rules:
  + The function will call **findAirlineRoute()** to see if the exact route with the origin, destination, and airline was already entered in the array. If it was found, then you will update the existing record in your array with the passenger data for that month. Recall there should be six entries (one for each month) for each route operated by an airline. If the route operated by the airline does not already exist in the array, add this new route to the array.
  + The function returns the actual number of **RouteRecord**’s used in the array. The value returned will be less than the size of the array created since not all records in the original CSV file will be entered into the array.
* **int findAirlineRoute( RouteRecord\* r, int length, const char\* origin, const char\* destination, const char\* airline, int curIdx )** – This **RECURSIVE** function finds a record in the **RouteRecord** array with the same origin and destination airport codes and airline. It returns the index number in which these three strings appear in the array. The function will return -1 if it cannot find these three strings in the same **struct** object.
* **void searchRecords( RouteRecord\* r, int length, const char\* key1, const char\* key2, SearchType st )** – This function searches the **RouteRecord** array and prints out the results of the search.
  + You will traverse the array and compare specific data members against the keys.
  + The parameter st determines if the function is searching by **ROUTE, ORIGIN, DESTINATION, AIRLINE**.
  + For **ORIGIN, DESTINATION, AIRLINE, key1** will hold the value you are looking for. For **ROUTE**, you are searching both the origin and destination and airport, so **key1** and **key2** will hold those values, respectively, that you will use to compare against the data members. For example, if the search is by the destination: **st** will be equal to **DESTINATION**, **key1** will have an airport code that the user entered, and you will compare each struct’s destination data member against the airport code.
  + You will print out the airline and the route for each matching value. Then, you will print out the total number of passengers on all matching records, total number of passengers by month for all matching records, as well as average numbers of passengers per month. Note that you must handle any instances where you the search has 0 results.
* **void printMenu()** – This function prints the menu. Here is the function below. Be sure to add this prototype to the header file.

**void printMenu()**

**{**

**printf( "\n\n######### Airline Route Records Database MENU #########\n" );**

**printf( "1. Search by Route\n" );**

**printf( "2. Search by Origin Airport\n" );**

**printf( "3. Search by Destination Airport\n" );**

**printf( "4. Search by Airline\n" );**

**printf( "5. Quit\n" );**

**printf( "Enter your selection: " );**

**}**

# TASK 3: Complete the main.c

* Download the attached **main.c**
* Follow the instructions written in the comments in the **main()** function.
* The **main()** is the driver of the program. It calls the functions above to load the data from the CSV file and to run queries that the user asks for.
* The name of the file will be passed in through **command line arguments**.
* The user will enter a numeric value from the menu. You must handle the case in which the user enters invalid values (e.g., strings).

# Task 4: Create a makefile

Create a **makefile** to compile and link all the files together. The grader will compile your code using your **makefile**.

# Submission

Be sure that your code follows the class coding style requirements. Your output should be similar in format as compared to the sample output shown below. Create a folder named your abc123, place all program files in this folder. Zip the folder and submit this abc123.zip file.

# Rubric

The rubric below is out of 50 points. However, the syllabus mentions that the project will be 40 points. The 10 points will be counted as extra credits. Your grade on Blackboard will be out of 40, but you can gain 50 / 40 if you solve the entire project correctly.

[6 points] – General Requirements

[2 points] – coding style - proper comments, indentation

[2 points] – correctness – compiles properly and gives correct output

[2 points] – submission – no missing files, zip, properly submitted

[3 points] - makefile

[3 points] - make compiles and creates the executable

[8 points] - main.c

[1 point] - Processes command-line argument for input file name

[1 point] - Open files and checks for success in opening

[2 points] - Sets up the array of RouteRecords

[2 points] - Processes user input, handling bad input

[2 points] - asks user for search keys and calls appropriate functions for desired functionality

[6 points] - route-records.h

[2 point] - header guard

[2 point] - enum and struct

[2 point] - function prototypes

[5 points] - createRecords()

[1 point] - Traverses the CSV file and counts the number of lines, excluding header row

[1 point] - Rewinds the file pointer

[1 point] - Dynamically allocates memory for the array of RouteRecords based on the number of lines counted

[1 point] - Initializes the passenger data array within each RouteRecord struct object

[1 point] - Returns the pointer to the array

[6 points] - fillRecords()

[1 point] - Traverses the CSV file skipping the header row

[1 point] - Calls findAirlineRoute() to see if route operated by airline already exists in the array

[1 point] - If found, updates an existing route operated by an airline in the array with the new month's passenger data

[1 point] - If not found, adds the new route operated by an airline into the array

[2 points] - Returns the unique number of routes as operated by different airlines. This will be the number of items in the array that are actually used.

[8 points] - findAirlineRoute()

[6 points] - **RECURSIVE** function set up to find the airline route by checking if there a struct object that matches the origin, destination, and airline

[1 point] - Returns the index of the array in which it was found

[1 point] - Returns -1 if not found

[8 points] - searchRecords()

[1 point] - Traverses the array

[1 point] - Determines the search type and compares key(s) against appropriate data members

[2 points] - Prints all matching information AirlineCode (Origin-Destination)

[4 points] - Calculates and prints out total passengers, total passengers per month, and average

# SAMPLE OUTPUT

Opening data.csv...

Unique routes operated by airlines: : 2763

######### Airline Route Records Database MENU #########

1. Search by Route

2. Search by Origin Airport

3. Search by Destination Airport

4. Search by Airline

5. Quit

Enter your selection: 1

Enter origin: LAX

Enter destination: LHR

Searching by route...

BA (LAX-LHR) EB (LAX-LHR) E9 (LAX-LHR) VS (LAX-LHR) UA (LAX-LHR) NZ (LAX-LHR) AA (LA X-LHR)

7 matches were found.

Statistics

Total Passengers: 323437

Total Passengers in Month 1: 127446

Total Passengers in Month 2: 111471

Total Passengers in Month 3: 72703

Total Passengers in Month 4: 4734

Total Passengers in Month 5: 3304

Total Passengers in Month 6: 3779

Average Passengers per Month: 53906

######### Airline Route Records Database MENU #########

1. Search by Route

2. Search by Origin Airport

3. Search by Destination Airport

4. Search by Airline

5. Quit

Enter your selection: 2

Enter origin: SFO

Search by origin...

CX (SFO-HKG) PR (SFO-CEB) CI (SFO-TPE) AS (SFO-PVR) AC (SFO-YYZ) LH (SFO-MUC) BA (SFO-LHR) NH (SFO-NRT) BR (SFO-TPE) PR (SFO-MNL) AC (SFO-YVR) OZ (SFO-ICN) KE (SFO-ICN) UA (SFO-FRA) UA (SFO-SYD) UA (SFO-NRT) AI (SFO-DEL) LF (SFO-FAB) UA (SFO-DEL) CZ (SFO-CAN) CZ (SFO-KMG) CZ (SFO-CGO) CZ (SFO-SJW) AI (SFO-BOM) CZ (SFO-SHE) CZ (SFO-TAO) UA (SFO-SJD) AS (SFO-SJD) UA (SFO-CUN) UA (SFO-PVR) MU (SFO-PVG) CA (SFO-NKG) UA (SFO-MEL) UA (SFO-ICN) QK (SFO-YVR) JL (SFO-HND) AM (SFO-GDL) BA (SFO-LGW) EB (SFO-BEG) OO (SFO-YVR) EB (SFO-CRK) FJ (SFO-NAN) QK (SFO-YEG) LY (SFO-TLV) IB (SFO-BCN) DY (SFO-BCN) QK (SFO-YYC) CA (SFO-PEK) DY (SFO-CDG) BF (SFO-PPT) UA (SFO-KIX) BF (SFO-ORY) TP (SFO-LIS) QF (SFO-BNE) QF (SFO-MEL) UA (SFO-CDG) OO (SFO-YYC) UA (SFO-YYZ) UA (SFO-PPT) SK (SFO-CPH) UA (SFO-HND) LX (SFO-ZRH) UA (SFO-TPE) DI (SFO-LGW) AC (SFO-YUL) LH (SFO-FRA) UA (SFO-MUC) TA (SFO-SAL) QF (SFO-SYD) EI (SFO-DUB) UA (SFO-MEX) UA (SFO-TLV) AF (SFO-CDG) AM (SFO-MEX) CM (SFO-PTY) UA (SFO-SIN) VS (SFO-LHR) KL (SFO-AMS) SQ (SFO-SIN) UA (SFO-LHR) UA (SFO-AKL) TK (SFO-IST) NZ (SFO-AKL) EK (SFO-DXB) UA (SFO-YVR) AN (SFO-YYC) UA (SFO-BOM) CZ (SFO-WUH) MU (SFO-TAO) UA (SFO-CTU) UA (SFO-BCN) UA (SFO-PEK) UA (SFO-PVG) UA (SFO-HKG) SQ (SFO-HKG) CX (SFO-NRT) FI (SFO-KEF) CX (SFO-KIX) 4O (SFO-GDL)

99 matches were found.

Statistics

Total Passengers: 2560583

Total Passengers in Month 1: 1183469

Total Passengers in Month 2: 840282

Total Passengers in Month 3: 433704

Total Passengers in Month 4: 44483

Total Passengers in Month 5: 17104

Total Passengers in Month 6: 41541

Average Passengers per Month: 426763

######### Airline Route Records Database MENU #########

1. Search by Route

2. Search by Origin Airport

3. Search by Destination Airport

4. Search by Airline

5. Quit

Enter your selection: 3

Enter destination: KIX

Searching by destination...

JL (LAX-KIX) UA (SFO-KIX) UA (GUM-KIX) DL (HNL-KIX) JL (HNL-KIX) HA (HNL-KIX) D7 (HNL-KIX) CX (ORD-KIX) CX (SFO-KIX)

9 matches were found.

Statistics

Total Passengers: 197847

Total Passengers in Month 1: 87720

Total Passengers in Month 2: 77050

Total Passengers in Month 3: 32397

Total Passengers in Month 4: 497

Total Passengers in Month 5: 0

Total Passengers in Month 6: 183

Average Passengers per Month: 32974

######### Airline Route Records Database MENU #########

1. Search by Route

2. Search by Origin Airport

3. Search by Destination Airport

4. Search by Airline

5. Quit

Enter your selection: 4

Enter airline: LO

Search by airline...

LO (ORD-WAW) LO (JFK-WAW) LO (IAD-WAW) LO (JFK-KIV) LO (JFK-BUD) LO (EWR-WAW) LO (LAX-WAW) LO (MIA-WAW) LO (ORD-KRK)

9 matches were found.

Statistics

Total Passengers: 111330

Total Passengers in Month 1: 48891

Total Passengers in Month 2: 36994

Total Passengers in Month 3: 20439

Total Passengers in Month 4: 787

Total Passengers in Month 5: 2450

Total Passengers in Month 6: 1769

Average Passengers per Month: 18555

######### Airline Route Records Database MENU #########

1. Search by Route

2. Search by Origin Airport

3. Search by Destination Airport

4. Search by Airline

5. Quit

Enter your selection: 10

Invalid choice.

######### Airline Route Records Database MENU #########

1. Search by Route

2. Search by Origin Airport

3. Search by Destination Airport

4. Search by Airline

5. Quit

Enter your selection: 5

Good-bye!