**Advanced Procedural Programming Assignment Project**

Name: Bernard Wong

ID: G00341962

# Summary

My design of the XYZ Airport Passenger Statistic System is more weight on user-friendly experience style. First, the program prompts a login.

After login the user can:

* Add a passenger to the database
* Display all passengers on the console
* Search and display a passenger details on the console
* Search and update a passenger’s details
* Search and delete a passenger from the database
* Generate statistics (based on criteria and region/trip duration)
* Output all passengers’ details and all criteria statistics onto a file
* List all passengers from UK in the order of year born
* Terminate program

After program terminates, the program updates the database for any changes.

Table of Contents

[Name: 1](#_Toc512180085)

[ID: 1](#_Toc512180086)

[Summary 1](#_Toc512180087)

[Code Documentations 4](#_Toc512180088)

[main.c 4](#_Toc512180089)

[Passenger.c 4](#_Toc512180090)

[Menu.c 4](#_Toc512180091)

[LinkedList.c 4](#_Toc512180092)

[Search.c 4](#_Toc512180093)

[Statistics.c 4](#_Toc512180094)

[FileHandling.c 4](#_Toc512180095)

[Passenger.h 4](#_Toc512180096)

[Linkedlist.h 4](#_Toc512180097)

[FileHandling.h 4](#_Toc512180098)

[Code 4](#_Toc512180099)

[Header Files 4](#_Toc512180100)

[Passenger.h 4](#_Toc512180101)

[Menu.h 4](#_Toc512180102)

[Linkedlist.h 4](#_Toc512180103)

[Search.h 4](#_Toc512180104)

[Statistic.h 4](#_Toc512180105)

[FileHandling.h 4](#_Toc512180106)

[Source Files 4](#_Toc512180107)

[main.c 4](#_Toc512180108)

[Passenger.c 4](#_Toc512180109)

[Menu.c 4](#_Toc512180110)

[Linkedlist.c 4](#_Toc512180111)

[Search.c 4](#_Toc512180112)

[Statistic.c 4](#_Toc512180113)

[FileHandling.c 4](#_Toc512180114)

[Text Files 4](#_Toc512180115)

[login.txt 4](#_Toc512180116)

[passenger.txt 4](#_Toc512180117)

[report.txt 4](#_Toc512180118)

[Screenshots 4](#_Toc512180119)

# Code Documentations

## main.c

* Where the main program executes
* The initialization of the linked list head pointer and tail pointer (headPtr & tailPtr)
* Prompt login
* Clears console and load the linked list from the “passenger.txt” file
* Main menu
* Before program terminates, update the “passenger.txt” file

## Passenger.c

* inputPassenger()
  + prompt user input every details of the new passenger
  + save all the details into newNode after inputs
* updateDetails()
  + prompt user input details of the passenger
  + the passport number/names and year born should not be change
  + overwrite the details of the passenger
* validatePassport()
  + check if the passport number exists in the database
  + returns 1 if the passport number is good to use
  + returns 0 if the passport number already exists
* validateEmail()
  + check if the input contains the symbol ‘@’ and ends with “.com”
  + loop through the input email string characters
  + if contains ‘@’ counter + 1
  + last 4 string index is “.com” counter + 1 for each character match
  + if every condition is met, the counter should increment to 5
  + returns 5 if email is good to use, if not 5, means that the email is invalid.
* getRegion()
* getTravelClass()
* getTripCount()
* getDuration()
  + the type of the data in passenger struct is int/char, so to output I need to map those integers/characters to a string
  + switch statement determines the output
  + this function takes the passenger data (int/char) and returns string

## Menu.c

* displayMenu()
  + display main menu and return user input choice
* displayHeader()
  + display the header of the program
* regionMenu()
* travelClassMenu()
* tripCountMenu()
* durationMenu()
  + display menu for prompt detail input and return user input
* statisticMenu()
  + display menu for prompt statistic criteria
  + if chosen criteria is travel class, call classCriteriaMenu()
  + if chosen criteria is Born Before 1980, set criteria to 5
  + return criteria
    - 1- economy
    - 2- premium economy
    - 3- business class
    - 4- first class
    - 5- born before 1980
* classCriteriaMenu()
  + display menu for prompt travel class statistic criteria and returns criteria
  + criteria 1 to 4 matches the data of passenger’s travel class integer 1 to 4, perfect for sorting and calculating statistics

## LinkedList.c

* addPassengerAtStart()
  + parse in new passenger node, head pointer and tail pointer
  + add the new node to the link list
  + set head and tail pointers point to the new node
* addPassenger()
  + where the main add passenger function executes
  + prompt input the new passenger details
  + if head pointer is null, call addPassengerAtStart()
  + while new passenger’s passport number is bigger than the current pointing passenger, iterate until it is not or until last passenger node
  + if pointing at last passenger node (means that new passenger has the biggest number) add to the end of the list
  + else should be pointing at a passenger in the list that is bigger than the new passenger, add before the bigger passenger
  + if the previous of the current pointing passenger is null, means that the new passenger is the smallest, change head pointer to new passenger
* displayList()
  + iterate through database and display every passenger’s details to the console
  + to output region/travelClass/tripCount/duration, call the string mapper (getters from Passenger.c) (or else it will output integers and characters)
* displayIndexPassenger()
  + parse in an index
  + loop through the list until counter equals to index
  + output every details of the passenger of that index
* listLength()
  + iterate and count the list
  + returns the counter (number of passengers)
* updatePassenger()
  + calls searchPassenger() from Search.c
  + if search fail (index = -1) do nothing
  + if found, set a temporary pointer to the index passenger
  + call updateDetails() from Passenger.c and parse in the temporary pointer
* deletePassenger()
  + calls searchPassenger() from Search.c
  + if search fail (index = -1) do nothing
  + if found, prompt user confirms the deletion since deletion is always important
  + if user wants to delete, set a temporary pointer to the index passenger
  + if only one passenger in the list, clear list
  + index = 0, delete first passenger of the list
  + index = last index of the list, delete last passenger of the list
  + if neither first nor last passenger, shift the list by let the index passenger’s previous node->next points at its next, index passenger’s next node->previous points at its previous
  + free the index passenger’s node
* sortedUKPassenger()
  + create a new linked list
  + iterate through the database
  + if the passenger is from UK, deep copy into a new node
  + iterate to the last node of the new list
  + if new list is empty add the new node to the list
  + if the new node is older than the youngest passenger, add before the youngest passenger
  + if the new node is younger than the passenger, add to the end of the new list
  + the list should be added nodes from oldest (smallest year born number) to youngest (biggest year born number) (sorted by year born)
  + display the new list by calling displayList() parsing in the new list

## Search.c

* searchPassenger()
  + prompt user chooses either search by passport number or full name
  + calls passportSearch() if search by passport number
  + calls nameSearch() if search by name
  + returns the index of the passenger found in the list
  + returns -1 if not found
* passportSearch()
  + prompt input passport number to search
  + iterate through the list
  + if the passport number matches one of the passengers, stop the loop and take the counter index of the loop
  + call displayIndexPassenger() from Linkedlist.c if found and returns the index
  + else output the passenger does not exist and returns -1
* nameSearch()
  + prompt input first name and second name (second name is necessary as first name may have duplicates)
  + iterate through the list
  + if the first name concat second name matches one of the passengers, stop the loop and take the counter index of the loop
  + call displayIndexPassenger() from Linkedlist.c if found and returns the index
  + else output the passenger does not exist and returns -1

## Statistics.c

* generateStatistics()
  + parse in an integer to determine if it should be output to a file or console
  + if console, take criteria (int) from statisticMenu() from Menu.c
  + switch statement map criteria to string for UI output heading
  + call calculateStatistics() and parse in criteria
* calculateStatistics()
  + if criteria is 1 to 4, call travelStatistics()
  + if criteria is 5, call bornBeforeStatistics()
* travelStatistics()
  + iterate through the list
  + because the criteria (1-4) is matched to the actual data travel class (1-4), can be used for determining if the passenger travel class matches the criteria
  + each statistic variable should have a counter individually
  + if match, each counter takes value from those counter incrementor functions and increment total counter
  + after the iterations, the function should have total number of (criteria) and each region/duration number of passenger
  + for loop 9 times (9 rows, 5 for each region, 4 for each duration), switch each loop, count = each counter and call displayStatistics() to display each row
* bornBeforeStatistics()
  + same as travelStatistics() instead matching criteria, condition is passenger year born is less than 1980
* fromUK()
* fromRoE()
* fromAsia()
* fromAmericas()
* fromAustralasia()
* avgOneDay()
* avgThreeDays()
* avgSevenDays()
* avgOverSeven()
  + return the counter if it does not match the data
  + increment the counter and return it if it match the data
* diisplayStatistics()
  + parse in the total number of passengers (based on criteria), the loop index and the count (based on the loop)
  + calculate the percentage = count / total \* 100
  + switch loop index map loop output string
  + if output to report file, parse in the file pointer and print into “report.txt” (append)
  + if output to console, output the row

## FileHandling.c

* initLinkedList()
* fopen file pointer to read “passenger.txt”
* read in all the passengers’ data into new node
* if head pointer is null, add the new node into as the first node
* because the passenger database is always sorted, so all other passengers after that only need to add to the end of the list
* loop until end of file
* there is a weird bug that always reads in the last empty new line, so getc() the new line and end the while loop and ungetc() back to the file
* updateLinkedList()
  + fopen file pointer to write “passenger.txt”
  + loop through all the passenger and overwrite every passenger detail into the file
* login()
  + prompt input username and password
  + for password need to mask the characters input into ‘\*’
  + do..while loop getch() character from the input, if the character is not return key or backspace key, save the character into the password string and printf an ‘\*’
  + if backspace key is pressed, decrement 1 to the counter of the password string
  + if there is no character, set counter back to 0, else printf ‘\b’ to delete an ‘\*’ from the console
  + fopen file pointer to read “login.txt”
  + compare the inputs and the text file data
  + if match, return 1
  + if not match return 0
* printReport()
  + fopen file pointer to write “report.txt”
  + loop through database and print all passenger details into the report file
  + close the file pointer and call printStatistics() to continue printing all criteria statistics after passengers
* printStatistics()
  + fopen file pointer to append “report.txt”
  + call each criteria 1 – 5 on calculateStatistics() from Statistic.c

## Passenger.h

* the main header file of the program
* import all the library needed for the whole program
* import all the header files of the project

## Linkedlist.h

* define struct passenger node
  + passport number (int)
  + first name (string)
  + second name (string)
  + year born (int)
  + email (string)
  + region (int)
  + travel class (int)
  + trip count (char)
  + duration (char)
  + next (struct passenger\*)
  + previous (struct passenger\*)

## FileHandling.h

* define constant LOGIN “login.txt”
* define constant PASSENGER “passenger.txt”
* define constant REPORT “report.txt”

# Code

## Header Files

### Passenger.h

// Passenger.h

#ifndef PASSENGER

#define PASSENGER

// import all the library needed for the whole project

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <windows.h>

// import all the header files of the project

#include "Linkedlist.h"

#include "Menu.h"

#include "Search.h"

#include "FileHandling.h"

#include "Statistic.h"

// function prototypes

void inputPassenger(struct passenger\* newNode, struct passenger\*\* top);

void updateDetails(struct passenger\* temp);

int validatePassport(int passportNum, struct passenger\*\* top);

int validateEmail(char email[30]);

char\* getRegion(int num);

char\* getTravelClass(int num);

char\* getTripCount(char num);

char\* getDuration(char num);

#endif // PASSENGER

### Menu.h

// Menu.h

#ifndef MENU

#define MENU

#include "Passenger.h"

// function prototypes

int displayMenu();

void displayHeader();

int regionMenu();

int travelClassMenu();

char tripCountMenu();

char durationMenu();

int statisticMenu();

int classCriteriaMenu();

#endif // MENU

### Linkedlist.h

// Linkedlist.h

#ifndef LINKEDLIST

#define LINKEDLIST

#include "Passenger.h"

// passenger struct

struct passenger {

int passportNum;

char firstName[20];

char secondName[20];

int yearBorn;

char email[30];

int region;

int travelClass;

char tripCount;

char duration;

struct passenger\* next;

struct passenger\* previous;

};

// function prototypes

void addPassengerAtStart(struct passenger\*\* top, struct passenger\*\* bottom, struct passenger\* newNode);

void addPassenger(struct passenger\*\* top, struct passenger\*\* bottom);

void displayList(struct passenger\* top);

void displayIndexPassenger(struct passenger\* top, int index);

int listLength(struct passenger\* top);

void updatePassenger(struct passenger\* top);

void deletePassenger(struct passenger\*\* top, struct passenger\*\* bottom);

void sortedUKPassenger(struct passenger\* top);

#endif // LINKEDLIST

### Search.h

// Search.h

#ifndef SEARCH

#define SEARCH

#include "Passenger.h"

// function prototypes

int searchPassenger(struct passenger\* top);

int passportSearch(struct passenger\* top);

int nameSearch(struct passenger\* top);

#endif // SEARCH

### Statistic.h

// Statistic.h

#ifndef STATISTICS

#define STATISTICS

#include "Passenger.h"

// function prototypes

void generateStatistics(struct passenger\* top, int report);

void calculateStatistics(int criteria, struct passenger\* top, int report, FILE\* fileptr);

void travelStatistics(int criteria, struct passenger\* top, int report, FILE\* fileptr);

void bornBeforeStatistics(struct passenger\* top, int report, FILE\* fileptr);

int fromUK(int ukCount, struct passenger\* temp);

int fromRoE(int RoECount, struct passenger\* temp);

int fromAsia(int asiaCount, struct passenger\* temp);

int fromAmericas(int americasCount, struct passenger\* temp);

int fromAustralasia(int ausCount, struct passenger\* temp);

int avgOneDay(int oneDay, struct passenger\* temp);

int avgThreeDays(int threeDays, struct passenger\* temp);

int avgSevenDays(int sevenDays, struct passenger\* temp);

int avgOverSeven(int overSevenDays, struct passenger\* temp);

void displayStatistics(int i, int count, int total, int report, FILE\* fileptr);

#endif // STATISTICS

### FileHandling.h

// FileHandling.h

#ifndef FILE\_HANDLING

#define FILE\_HANDLING

#include "Passenger.h"

// constants

#define LOGIN "login.txt"

#define PASSENGER "passenger.txt"

#define REPORT "report.txt"

// function prototypes

void initLinkedList(struct passenger\*\* top, struct passenger\*\* bottom);

void updateLinkedList(struct passenger\* top);

int login();

void printReport(struct passenger\* top);

void printStatistics(struct passenger\* top);

#endif // FILE\_HANDLING

## Source Files

### main.c

// Main.c

#include "Passenger.h"

void main() {

struct passenger\* headPtr = NULL;

struct passenger\* tailPtr = NULL;

int choice;

int position;

int log;

// login

displayHeader();

log = login();

while (log != 1) {

system("@cls||clear");

displayHeader();

printf("The Username Or Password is Incorrect.\n");

log = login();

} // while invalid login

system("@cls||clear");

displayHeader();

initLinkedList(&headPtr, &tailPtr); // Load linked list from file

Sleep(500);

system("@cls||clear");

// main menu

displayHeader();

choice = displayMenu();

while (choice != -1) {

switch (choice) {

case 1:

addPassenger(&headPtr, &tailPtr);

break;

case 2:

if (headPtr == NULL) {

printf("\nThe database is empty.\n");

}

else {

displayList(headPtr);

} // if linkedlist is empty

break;

case 3:

searchPassenger(headPtr);

break;

case 4:

updatePassenger(headPtr);

break;

case 5:

deletePassenger(&headPtr, &tailPtr);

break;

case 6:

generateStatistics(headPtr, 0);

break;

case 7:

printReport(headPtr);

break;

case 8:

if (headPtr == NULL) {

printf("\nThe database is empty.\n");

}

else {

sortedUKPassenger(headPtr);

} // if linkedlist is empty

break;

default:

printf("Please enter 1 to 8 only or -1 to exit.\n"); // validation

} // switch main menu user input choice

choice = displayMenu();

} // while -1 sentinel control

updateLinkedList(headPtr); // update linked list when program terminated

printf("\n==Program Terminated==\n");

getch();

} // main()

### Passenger.c

// Passenger.c

#include "Passenger.h"

// prompt user input every details of the new passenger

void inputPassenger(struct passenger\* newNode, struct passenger\*\* top) {

int passportNum;

int passportValidate = 1;

char emailAddress[30];

int emailValidate = 0;

printf("\nPlease enter the following:\n");

printf("Passport Number: ");

scanf("%d", &passportNum);

// if headpointer is not NULL

if (\*top != NULL)

passportValidate = validatePassport(passportNum, top);

while (passportValidate != 1) {

printf("Passport number already exists in the database, please try again.\n");

printf("Passport Number: ");

scanf("%d", &passportNum);

passportValidate = validatePassport(passportNum, top);

} // while passport validation

newNode->passportNum = passportNum;

printf("First Name: ");

scanf("%s", newNode->firstName);

printf("Second Name: ");

scanf("%s", newNode->secondName);

printf("Year Born: ");

scanf("%d", &newNode->yearBorn);

printf("Email Address: ");

scanf("%s", emailAddress);

emailValidate = validateEmail(emailAddress);

while (emailValidate != 5) {

printf("Invalid email address, must contain an '@' and '.com'\n");

printf("Email Address: ");

scanf("%s", emailAddress);

emailValidate = validateEmail(emailAddress);

} // while email validation

strcpy(newNode->email, emailAddress);

newNode->region = regionMenu();

newNode->travelClass = travelClassMenu();

newNode->tripCount = tripCountMenu();

newNode->duration = durationMenu();

} // inputPassenger()

// prompt user update the details of the passenger

void updateDetails(struct passenger\* temp) {

char emailAddress[30];

int emailValidate = 0;

printf("\nPlease enter the following:\n");

printf("Email Address: ");

scanf("%s", emailAddress);

emailValidate = validateEmail(emailAddress);

while (emailValidate != 5) {

printf("Invalid email address, must contain an '@' and '.com'\n");

printf("Email Address: ");

scanf("%s", emailAddress);

emailValidate = validateEmail(emailAddress);

} // while email validation

strcpy(temp->email, emailAddress);

temp->region = regionMenu();

temp->travelClass = travelClassMenu();

temp->tripCount = tripCountMenu();

temp->duration = durationMenu();

} //updateDetails()

// check if the passport number is already exists in the database (return 1 if did not exists or 0 if exists)

int validatePassport(int passportNum, struct passenger\*\* top) {

struct passenger\* temp;

temp = \*top;

while (temp != NULL) {

// return 0 if the passport number already exists

if (passportNum == temp->passportNum)

return 0;

temp = temp->next;

} // while iterate through linked list

return 1;

} // validatePassport()

// check if the input contains the symbol '@' and ends with ".com" (return 5 if the email is good)

int validateEmail(char email[30]) {

int length = strlen(email);

int valid = 0;

for (int i = 0; i < length; i++) {

if (email[i] == '@') {

valid++;

break;

} // if one of the character is '@' symbol

} // for i

// if ends with ".com"

if (email[length - 4] == '.')

valid++;

if (email[length - 3] == 'c')

valid++;

if (email[length - 2] == 'o')

valid++;

if (email[length - 1] == 'm')

valid++;

// valid counter should increment to 5 if each condition is met, else the email cannot be use

return valid;

} // validateEmail()

// return the string of the region (it is an integer in the database)

char\* getRegion(int num) {

char region[15];

switch (num) {

case 1:

strcpy(region, "UK");

break;

case 2:

strcpy(region, "Rest of Europe");

break;

case 3:

strcpy(region, "Asia");

break;

case 4:

strcpy(region, "Americas");

break;

case 5:

strcpy(region, "Australasia");

break;

default:

strcpy(region, "Others"); // debug purpose

} // switch num

return region;

} // getRegion()

// return the string of the travel class (it is an integer in the database)

char\* getTravelClass(int num) {

char travelClass[16];

switch (num) {

case 1:

strcpy(travelClass, "Economy");

break;

case 2:

strcpy(travelClass, "Premium Economy");

break;

case 3:

strcpy(travelClass, "Business Class");

break;

case 4:

strcpy(travelClass, "First Class");

break;

default:

strcpy(travelClass, "Others"); // debug purpose

}

return travelClass;

} // getTravelClass()

// return the string of the trip count (it is a character in the database)

char\* getTripCount(char num) {

char tripCount[32];

switch (num) {

case 'a':

strcpy(tripCount, "Less than three times per year");

break;

case 'b':

strcpy(tripCount, "Less than five times per year");

break;

case 'c':

strcpy(tripCount, "More than five times per year");

break;

default:

strcpy(tripCount, "Others"); // debug purpose

} // switch num

return tripCount;

} // getTripCount()

// return the string of the duration (it is a character in the database)

char\* getDuration(char num) {

char duration[18];

switch (num) {

case 'a':

strcpy(duration, "One day");

break;

case 'b':

strcpy(duration, "Less than 3 days");

break;

case 'c':

strcpy(duration, "Less than 7 days");

break;

case 'd':

strcpy(duration, "More than 7 days");

break;

default:

strcpy(duration, "Others"); // debug purpose

} // switch num

return duration;

} // getDuration()

### Menu.c

// Menu.c

#include "Passenger.h"

// display main menu and return user input choice

int displayMenu() {

int choice;

printf("\n+-------------------------------------------------------------+\n");

printf("|1. Add a passenger |\n");

printf("|2. Display all passenger |\n");

printf("|3. Display passenger details |\n");

printf("|4. Update a passenger statistic |\n");

printf("|5. Delete a passenger |\n");

printf("|6. Generate statistics |\n");

printf("|7. Print all passenger details into a report file |\n");

printf("|8. List all the passenger from UK in order of year born |\n");

printf("|-1. Exit |\n");

printf("+-------------------------------------------------------------+");

printf("\nPlease enter one of the following options: ");

scanf("%d", &choice);

return choice;

} // displayMenu()

// display the header of the program

void displayHeader() {

printf("+============================================================+\n");

printf("\* \*\n");

printf("\* Welcome to XYZ Airport Passenger Statistic System \*\n");

printf("\* \*\n");

printf("+============================================================+\n");

} // displayHeader()

// display menu for prompt region input and return user input

int regionMenu() {

int region;

printf("Which of the following areas did you travel from (1-5):\n");

printf("\t(1) UK\n");

printf("\t(2) Rest of Europe\n");

printf("\t(3) Asia\n");

printf("\t(4) Americas\n");

printf("\t(5) Australasia\n");

printf("Please enter your region: ");

scanf("%d", &region);

while(region < 1 || region > 5) {

printf("\nInvalid input, 1 to 5 only.\n");

printf("Please enter your region: ");

scanf("%d", &region);

} // while validate

return region;

} // regionMenu()

// display menu for prompt travel class input and return user input

int travelClassMenu() {

int travelClass;

printf("Which of the following travel class did you use to travel to Ireland (1-4):\n");

printf("\t(1) Economy\n");

printf("\t(2) Premium Economy\n");

printf("\t(3) Business Class\n");

printf("\t(4) First Class\n");

printf("Please enter your travel class: ");

scanf("%d", &travelClass);

while(travelClass < 1 || travelClass > 4) {

printf("\nInvalid input, 1 to 4 only.\n");

printf("Please enter your travel class: ");

scanf("%d", &travelClass);

} // while validate

return travelClass;

} // travelClassMenu()

// display menu for prompt trip count input and return user input

char tripCountMenu() {

char travelClass;

printf("How many trips to Ireland do you make per year (a/b/c):\n");

printf("\t(a) Less than three times per year\n");

printf("\t(b) Less than five times per year\n");

printf("\t(c) More than five times per year\n");

printf("Please enter a / b / c: ");

scanf(" %c", &travelClass);

while(!(travelClass == 'a' || travelClass == 'b' || travelClass == 'c')) {

printf("\nInvalid input, a / b / c only.\n");

printf("Please enter a / b / c: ");

scanf(" %c", &travelClass);

} // while validate

return travelClass;

} // tripCountMenu()

// display menu for prompt trip duration input and return user input

char durationMenu() {

char duration;

printf("On average how long is your duration (a/b/c/d):\n");

printf("\t(a) One day\n");

printf("\t(b) Less than 3 days\n");

printf("\t(c) Less than 7 day\n");

printf("\t(d) More than 7 day\n");

printf("Please enter a / b / c / d: ");

scanf(" %c", &duration);

while(!(duration == 'a' || duration == 'b' || duration == 'c' || duration == 'd')) {

printf("\nInvalid input, a / b / c / d only.\n");

printf("Please enter a / b / c / d: ");

scanf(" %c", &duration);

} // while validate

return duration;

} // durationMenu()

// display menu for prompt statistic criteria and return criteria

int statisticMenu() {

int criteria;

printf("\nWhich of the following criteria do you wish to generate statistics from (1-2): \n");

printf("\t(1) Travel Class\n");

printf("\t(2) Born Before 1980\n");

printf("Please enter criteria: ");

scanf("%d", &criteria);

while (!(criteria == 1 || criteria == 2)) {

printf("\nInvalid input, 1 and 2 only.\n");

printf("Please enter criteria: ");

scanf("%d", &criteria);

} // while validate

// call classCriteriaMenu for 1 and set criteria = 5 for born before 1980

if (criteria == 1)

criteria = classCriteriaMenu();

else

criteria = 5;

return criteria;

} // statisticMenu()

// display menu for prompt travel class statistic criteria and return criteria

int classCriteriaMenu() {

int classCriteria;

printf("\nWhich travel class do you wish to generate statistics from (1-4):\n");

printf("\t(1) Economy\n");

printf("\t(2) Premium Economy\n");

printf("\t(3) Business Class\n");

printf("\t(4) First Class\n");

printf("Please enter travel class criteria: ");

scanf("%d", &classCriteria);

while (classCriteria < 1 || classCriteria > 4) {

printf("\nInvalid input, 1 to 4 only.\n");

printf("Please enter criteria: ");

scanf("%d", &classCriteria);

} // while validate

return classCriteria;

} // classCriteriaMenu()

### Linkedlist.c

// Linkedlist.c

#include "Linkedlist.h"

// add the new passenger node to the start of the linked list

void addPassengerAtStart(struct passenger\*\* top, struct passenger\*\* bottom, struct passenger\* newNode) {

newNode->next = NULL;

newNode->previous = NULL;

\*top = newNode;

\*bottom = newNode;

} // addPassengerAtStart()

// add the passenger by comparing the passport number so the linked list is in order

void addPassenger(struct passenger\*\* top, struct passenger\*\* bottom) {

struct passenger\* newNode;

struct passenger\* temp;

struct passenger\* temp2;

newNode = (struct passenger\*)malloc(sizeof(struct passenger));

inputPassenger(newNode, top);

if (\*top == NULL) {

addPassengerAtStart(top, bottom, newNode);

}

else {

temp = \*top;

while (newNode->passportNum > temp->passportNum) {

temp2 = temp;

temp = temp->next;

if (temp == NULL)

break;

} // while new passenger's passport number is bigger than current pointing passenger

if (temp == NULL) {

// add to the end of the linked list

newNode->next = NULL;

newNode->previous = temp2;

temp2->next = newNode;

\*bottom = newNode;

}

else {

// add before the current pointing passenger

newNode->next = temp;

newNode->previous = temp->previous;

temp2 = temp->previous;

if (temp2 == NULL) {

\*top = newNode;

}

else {

temp2->next = newNode;

} // if the previous of the current pointing passenger is NULL

temp->previous = newNode;

} // if current pointing passenger is NULL

printf("Passenger Successfully Added.\n");

} // if head pointer is NULL

} // addPassenger()

// iterate through database and display every passenger's details to the console

void displayList(struct passenger\* top) {

struct passenger\* temp;

int passengerCount = 1;

char region[15];

char travelClass[16];

char tripCount[32];

char duration[18];

temp = top;

printf("\nPassenger Database:");

while (temp != NULL) {

// output every details

printf("\n==Passenger %d==\n", passengerCount);

printf("Passport Number: %d\n", temp->passportNum);

printf("First Name: %s\n", temp->firstName);

printf("Second Name: %s\n", temp->secondName);

printf("Year Born: %d\n", temp->yearBorn);

printf("Email Address: %s\n", temp->email);

strcpy(region, getRegion(temp->region));

strcpy(travelClass, getTravelClass(temp->travelClass));

strcpy(tripCount, getTripCount(temp->tripCount));

strcpy(duration, getDuration(temp->duration));

printf("Region: %s\n", region);

printf("Travel Class: %s\n", travelClass);

printf("Trip Count to Ireland: %s\n", tripCount);

printf("Stay Duration: %s\n", duration);

printf("================\n");

temp = temp->next;

passengerCount++;

} // while current pointing passenger is not NULL

} // displayList()

// display the passenger details that is at the provided index

void displayIndexPassenger(struct passenger\* top, int index) {

struct passenger\* temp;

int count = 0;

char region[15];

char travelClass[16];

char tripCount[32];

char duration[18];

temp = top;

while (count != index) {

count++;

temp = temp->next;

} // while counter is not equals to index

// output every details

printf("\n==Passenger %d==\n", index + 1);

printf("Passport Number: %d\n", temp->passportNum);

printf("First Name: %s\n", temp->firstName);

printf("Second Name: %s\n", temp->secondName);

printf("Year Born: %d\n", temp->yearBorn);

printf("Email Address: %s\n", temp->email);

strcpy(region, getRegion(temp->region));

strcpy(travelClass, getTravelClass(temp->travelClass));

strcpy(tripCount, getTripCount(temp->tripCount));

strcpy(duration, getDuration(temp->duration));

printf("Region: %s\n", region);

printf("Travel Class: %s\n", travelClass);

printf("Trip Count to Ireland: %s\n", tripCount);

printf("Stay Duration: %s\n", duration);

printf("================\n");

} // displayIndexPassenger()

// return the length of the linked list

int listLength(struct passenger\* top) {

struct passenger\* temp;

int count = 0;

temp = top;

while (temp != NULL) {

count++;

temp = temp->next;

} // while current pointing counter is not NULL

return count;

} // listLength

// search and update the passenger

void updatePassenger(struct passenger\* top) {

struct passenger\* temp;

int count = 0;

int index;

index = searchPassenger(top);

temp = top;

// search fail do nothing

if (index != -1) {

while (count != index) {

temp = temp->next;

count++;

} // while count not equals index

updateDetails(temp);

} // if index not equals -1

} // updatePassenger()

// search and delete the passenger

void deletePassenger(struct passenger\*\* top, struct passenger\*\* bottom) {

struct passenger\* temp;

struct passenger\* temp2;

int count = 0;

int index;

char choice;

index = searchPassenger(\*top);

temp = \*top;

// search fail do nothing

if (index != -1) {

printf("\nAre you sure you want to delete this passenger? (y/n): ");

scanf(" %c", &choice);

while (!(choice == 'y' || choice == 'n')) {

printf("Invalid input, y or n only.\n");

printf("Please enter your choice: ");

scanf(" %c", &choice);

}

if (choice == 'y') {

while (count != index) {

temp = temp->next;

count++;

}

if ((\*top)->next == NULL) {

// if only one passenger in the list, clear list

\*top = NULL;

\*bottom = NULL;

}

else if (index == 0) {

// delete first passenger of the list

\*top = temp->next;

(\*top)->previous = NULL;

}

else if (index == listLength(\*top) - 1) {

// delete last passenger of the list

\*bottom = temp->previous;

(\*bottom)->next = NULL;

}

else {

// if neither first nor last passenger, shift and delete

temp2 = temp->previous;

temp2->next = temp->next;

temp2 = temp->next;

temp2->previous = temp->previous;

} // if/else if

free(temp); // free memory

printf("\nPassenger Successfully Deleted.\n");

}

else {

printf("\nDeletion Canceled.\n");

} // if choice is y

} // if index not equals -1

} // deletePassenger()

// create new linked list and add all uk passenger into the new list sorted by year born

void sortedUKPassenger(struct passenger\* top) {

struct passenger\* tempHead = NULL;

struct passenger\* newNode;

struct passenger\* temp;

struct passenger\* temp2;

struct passenger\* temp3;

temp = top;

for (int i = 0; i < listLength(top); i++) {

newNode = (struct passenger\*)malloc(sizeof(struct passenger));

temp2 = tempHead;

if (temp->region == 1) {

// deep copy the passenger into a new passenger node

newNode->passportNum = temp->passportNum;

strcpy(newNode->firstName, temp->firstName);

strcpy(newNode->secondName, temp->secondName);

newNode->yearBorn = temp->yearBorn;

strcpy(newNode->email, temp->email);

newNode->region = temp->region;

newNode->travelClass = temp->travelClass;

newNode->tripCount = temp->tripCount;

newNode->duration = temp->duration;

newNode->previous = NULL;

if (tempHead == NULL) {

// if new list is empty

tempHead = newNode;

newNode->next = NULL;

}

else {

if (newNode->yearBorn >= temp2->yearBorn) {

while (temp2->next != NULL && newNode->yearBorn >= temp2->yearBorn) {

// point to the biggest year born (youngest) passenger

temp2 = temp2->next;

} // while current passenger's year born is bigger (younger) than or equals current pointing passenger (next pointer must not be NULL)

if (newNode->yearBorn < temp2->yearBorn) {

// if the current passenger's year born is smaller (older) than the youngest passenger, add before the youngest passenger

newNode->next = temp2;

temp3 = temp2->previous;

temp2->previous = newNode;

newNode->previous = temp3;

temp3->next = newNode;

}

else {

// if the current passenger's year born is bigger (younger) than the youngest passenger, add to the end of the list

newNode->next = temp2->next;

newNode->previous = temp2;

temp2->next = newNode;

} // if current passenger's year born is smaller (older) than current pointing passenger

}

else {

newNode->next = temp2;

tempHead = newNode;

} // if current passenger's year born is bigger (younger) than or equals current pointing passenger

} // if current pointing passenger is NULL (new list)

} // if region is UK

temp = temp->next;

} // for i

displayList(tempHead); // display the new list

} // sortedUKPassenger()

### Search.c

// Search.c

#include "Search.h"

// search the passenger using either passport number of full name (return the index of the passenger found in the list or -1 if not found)

int searchPassenger(struct passenger\* top) {

int choice;

int index;

printf("\nWhich do you wish to search by:\n");

printf("(1) Passport Number\n");

printf("(2) Full Name\n");

printf("Please enter your choice: ");

scanf("%d", &choice);

while (!(choice == 1 || choice == 2)) {

printf("Invalid input, 1 or 2 only.\n");

printf("Please enter your choice: ");

scanf(" %d", &choice);

} // while validation

if (choice == 1) {

index = passportSearch(top);

}

else {

index = nameSearch(top);

} // if determine using passport number or name for searching

return index;

} // searchPassenger()

// search passenger by passport number (return the index if found and -1 if not found)

int passportSearch(struct passenger\* top) {

struct passenger\* temp;

int passportNum;

int found = 0;

int count = 0;

temp = top;

printf("\nPlease enter passport number: ");

scanf("%d", &passportNum);

for (int i = 0; i < listLength(top); i++) {

if (temp->passportNum == passportNum) {

// stop the loop

i = listLength(top);

found = 1;

}

else {

// continue the iteration

temp = temp->next;

count++;

} // if passport number matches

} // for iterate through the list

if (found == 1) {

displayIndexPassenger(top, count);

return count;

}

else {

printf("The passenger does not exist in the database.\n");

} // if found passenger

return -1;

} // passportSearch()

// search passenger by first and second name (return the index if found and -1 if not found)

int nameSearch(struct passenger\* top) {

struct passenger\* temp;

char sFirstName[20];

char sSecondName[20];

char sFullName[40];

char firstName[20];

char secondName[20];

char fullName[40];

int found = 0;

int count = 0;

temp = top;

printf("\nPlease enter first name: ");

scanf("%s", sFirstName);

printf("Please enter second name: ");

scanf("%s", sSecondName);

strcpy(sFullName, strcat(sFirstName, sSecondName));

for (int i = 0; i < listLength(top); i++) {

strcpy(firstName, temp->firstName);

strcpy(secondName, temp->secondName);

strcpy(fullName, strcat(firstName, secondName));

if (strcmp(fullName, sFullName) == 0) {

// stop the loop

found = 1;

i = listLength(top);

}

else {

// continue the iteration

temp = temp->next;

count++;

} // if full name match

} // for iterate through the list

if (found == 1) {

displayIndexPassenger(top, count);

return count;

}

else {

printf("The passenger does not exist in the database.\n");

} // if passenger found

return -1;

} // nameSearch()

### Statistic.c

// Statistic.c

#include "Statistic.h"

// generate statistic and either output to console or print into "report.txt"

void generateStatistics(struct passenger\* top, int report) {

int criteria;

char s[18];

if (report == 0) {

FILE\* fileptr = NULL;

criteria = statisticMenu();

switch (criteria) {

case 1:

strcpy(s, "Economy");

break;

case 2:

strcpy(s, "Premium Economy");

break;

case 3:

strcpy(s, "Business Class");

break;

case 4:

strcpy(s, "First Class");

break;

case 5:

strcpy(s, "Born Before 1980");

break;

default:

printf("Error outputting criteria.\n"); // debug purpose

} // switch criteria

printf("\n==========Statistics From Criteria \"%s\"==========\n", s);

calculateStatistics(criteria, top, report, fileptr);

}

else {

printStatistics(top);

} // if determine print to report or console

} // generateStatistics()

// calculate statistics base on criteria

void calculateStatistics(int criteria, struct passenger\* top, int report, FILE\* fileptr) {

// if criteria is 5, criteria is "born beofer 1980"

if (criteria >= 1 && criteria <= 4)

travelStatistics(criteria, top, report, fileptr);

else

bornBeforeStatistics(top, report, fileptr);

} // calculateStatistics()

// calculate travel class statistics

void travelStatistics(int criteria, struct passenger\* top, int report, FILE\* fileptr) {

struct passenger\* temp;

int total = 0;

int count;

int ukCount = 0, RoECount = 0, asiaCount = 0, americasCount = 0, ausCount = 0;

int oneDay = 0, threeDays = 0, sevenDays = 0, overSevenDays = 0;

temp = top;

for (int i = 0; i < listLength(top); i++) {

if (temp->travelClass == criteria) {

// count the passenger count needed for the statistics

ukCount = fromUK(ukCount, temp);

RoECount = fromRoE(RoECount, temp);

asiaCount = fromAsia(asiaCount, temp);

americasCount = fromAmericas(americasCount, temp);

ausCount = fromAustralasia(ausCount, temp);

oneDay = avgOneDay(oneDay, temp);

threeDays = avgThreeDays(threeDays, temp);

sevenDays = avgSevenDays(sevenDays, temp);

overSevenDays = avgOverSeven(overSevenDays, temp);

total++;

} // if criteria has same integer as travel class

temp = temp->next;

} // for loop through the list

for (int i = 0; i < 9; i++) {

switch (i) {

case 0:

count = ukCount;

break;

case 1:

count = RoECount;

break;

case 2:

count = asiaCount;

break;

case 3:

count = americasCount;

break;

case 4:

count = ausCount;

break;

case 5:

count = oneDay;

break;

case 6:

count = threeDays;

break;

case 7:

count = sevenDays;

break;

case 8:

count = overSevenDays;

break;

default:

printf("Error in calculating statistics.\n"); // debug purpose

} // switch i for rows

displayStatistics(i, count, total, report, fileptr);

} // for loop 9 rows of statistics

} // travelStatistics()

void bornBeforeStatistics(struct passenger\* top, int report, FILE\* fileptr) {

struct passenger\* temp;

int total = 0;

int count;

int ukCount = 0, RoECount = 0, asiaCount = 0, americasCount = 0, ausCount = 0;

int oneDay = 0, threeDays = 0, sevenDays = 0, overSevenDays = 0;

temp = top;

for (int i = 0; i < listLength(top); i++) {

if (temp->yearBorn < 1980) {

// count the passenger count needed for the statistics

ukCount = fromUK(ukCount, temp);

RoECount = fromRoE(RoECount, temp);

asiaCount = fromAsia(asiaCount, temp);

americasCount = fromAmericas(americasCount, temp);

ausCount = fromAustralasia(ausCount, temp);

oneDay = avgOneDay(oneDay, temp);

threeDays = avgThreeDays(threeDays, temp);

sevenDays = avgSevenDays(sevenDays, temp);

overSevenDays = avgOverSeven(overSevenDays, temp);

total++;

} // if passenger born before 1980

temp = temp->next;

} // for loop through the list

for (int i = 0; i < 9; i++) {

switch (i) {

case 0:

count = ukCount;

break;

case 1:

count = RoECount;

break;

case 2:

count = asiaCount;

break;

case 3:

count = americasCount;

break;

case 4:

count = ausCount;

break;

case 5:

count = oneDay;

break;

case 6:

count = threeDays;

break;

case 7:

count = sevenDays;

break;

case 8:

count = overSevenDays;

break;

default:

printf("Error in calculating statistics.\n"); // debug purpose

} // switch i for rows

displayStatistics(i, count, total, report, fileptr);

} // for loop 9 rows of statistics

} // bornBeforeStatistics()

// if passenger is from UK, return incremented counter or just return the counter if is not

int fromUK(int ukCount, struct passenger\* temp) {

if (temp->region == 1) {

return ++ukCount;

} // if

return ukCount;

} // fromUK()

// if passenger is from Rest of Europe, return incremented counter or just return the counter if is not

int fromRoE(int RoECount, struct passenger\* temp) {

if (temp->region == 2) {

return ++RoECount;

} // if

return RoECount;

} // fromRoE()

// if passenger is from Asia, return incremented counter or just return the counter if is not

int fromAsia(int asiaCount, struct passenger\* temp) {

if (temp->region == 3) {

return ++asiaCount;

} // if

return asiaCount;

} // fromAsia()

// if passenger is from Americas, return incremented counter or just return the counter if is not

int fromAmericas(int americasCount, struct passenger\* temp) {

if (temp->region == 4) {

return ++americasCount;

} // if

return americasCount;

} // fromAmericas()

// if passenger is from UK, return incremented counter or just return the counter if is not

int fromAustralasia(int ausCount, struct passenger\* temp) {

if (temp->region == 5) {

return ++ausCount;

} // if

return ausCount;

} // fromAustralasia()

// if passenger is staying one day, return incremented counter or just return the counter if is not

int avgOneDay(int oneDay, struct passenger\* temp) {

if (temp->duration == 'a') {

return ++oneDay;

} // if

return oneDay;

} // avgOneDay()

// if passenger is staying average 3 days, return incremented counter or just return the counter if is not

int avgThreeDays(int threeDays, struct passenger\* temp) {

if (temp->duration == 'b') {

return ++threeDays;

} // if

return threeDays;

} // avgThreeDays()

// if passenger is staying average 7 days, return incremented counter or just return the counter if is not

int avgSevenDays(int sevenDays, struct passenger\* temp) {

if (temp->duration == 'c') {

return ++sevenDays;

} // if

return sevenDays;

} // avgSevenDays()

// if passenger is staying over 7 days, return incremented counter or just return the counter if is not

int avgOverSeven(int overSevenDays, struct passenger\* temp) {

if (temp->duration == 'd') {

return ++overSevenDays;

} // if

return overSevenDays;

} // avgOverSeven()

// actual calculation and display the statistics

void displayStatistics(int i, int count, int total, int report, FILE\* fileptr) {

char s[45];

float percentage;

// percentage calculation

percentage = (float)count / total \* 100;

switch (i) {

case 0:

strcpy(s, "travel from the UK");

break;

case 1:

strcpy(s, "travel from the Rest of Europe");

break;

case 2:

strcpy(s, "travel from the Asia");

break;

case 3:

strcpy(s, "travel from the Americas");

break;

case 4:

strcpy(s, "travel from the Australasia");

break;

case 5:

strcpy(s, "spent on average one day in Ireland");

break;

case 6:

strcpy(s, "spent on average less than 3 days in Ireland");

break;

case 7:

strcpy(s, "spent on average less than 7 days in Ireland");

break;

case 8:

strcpy(s, "spent on average more than 7 days in Ireland");

break;

default:

printf("Error in outputting statistics.\n"); // debug purpose

} // switch i for rows

if (report == 1) {

if (fileptr == NULL)

{

// output error

printf("!- Warning -!\n");

printf("!- The File \"report.txt\" Does Not Exist -!\n");

printf("!- Warning -!\n");

}

else {

fprintf(fileptr, "Percentage of passengers who %s : %.2f%%\n", s, percentage);

} // if file pointer is NULL

}

else {

printf("Percentage of passengers who %s : %.2f%%\n", s, percentage);

} // if determine print to report file or console

} // displayStatistics()

### FileHandling.c

// FileHandling.c

#include "FileHandling.h"

// load linked list from "passenger.txt"

void initLinkedList(struct passenger\*\* top, struct passenger\*\* bottom) {

FILE\* fileptr;

struct passenger\* newNode;

struct passenger\* temp;

fileptr = fopen(PASSENGER, "r");

char c;

if (fileptr == NULL) {

// output error

printf("!- Warning -!\n");

printf("!- The File \"passenger.txt\" Does Not Exist -!\n");

printf("!- Warning -!\n");

}

else {

// useless loading console output (user experience)

printf("Loading Data From Database");

Sleep(250);

printf(".");

Sleep(250);

printf(".");

Sleep(250);

printf(".");

Sleep(250);

printf("\n");

while (!feof(fileptr)) {

newNode = (struct passenger\*)malloc(sizeof(struct passenger));

// load the passengers' details into the list

fscanf(fileptr, "%d", &newNode->passportNum);

fscanf(fileptr, "%s", newNode->firstName);

fscanf(fileptr, "%s", newNode->secondName);

fscanf(fileptr, "%d", &newNode->yearBorn);

fscanf(fileptr, "%s", newNode->email);

fscanf(fileptr, "%d", &newNode->region);

fscanf(fileptr, "%d", &newNode->travelClass);

fscanf(fileptr, " %c", &newNode->tripCount);

fscanf(fileptr, " %c", &newNode->duration);

temp = \*bottom;

if (\*top == NULL) {

// first passenger node

newNode->next = NULL;

newNode->previous = NULL;

\*top = newNode;

\*bottom = newNode;

}

else {

// because it is always sorted, only need to add to the end of the list

newNode->next = NULL;

newNode->previous = temp;

temp->next = newNode;

\*bottom = newNode;

} // if head pointer is NULL

// file pointer always reads in the last empty line/ getch the empty line and break

if ((c = fgetc(fileptr)) == EOF || (c = fgetc(fileptr)) == '\n')

break;

ungetc(c, fileptr);

} // while not end of file

printf("Loading Complete.");

fclose(fileptr);

} // if file pointer is NULL

} // initLinkedList()

// update the linked list to "passenger.txt" when user terminate program

void updateLinkedList(struct passenger\* top) {

FILE\* fileptr;

struct passenger\* temp;

fileptr = fopen(PASSENGER, "w");

temp = top;

if (fileptr == NULL) {

// output error

printf("!- Warning -!\n");

printf("!- The File \"passenger.txt\" Does Not Exist -!\n");

printf("!- Warning -!\n");

}

else {

while (temp != NULL) {

// print passenger's details to the file

fprintf(fileptr, "%d\n", temp->passportNum);

fprintf(fileptr, "%s\n", temp->firstName);

fprintf(fileptr, "%s\n", temp->secondName);

fprintf(fileptr, "%d\n", temp->yearBorn);

fprintf(fileptr, "%s\n", temp->email);

fprintf(fileptr, "%d\n", temp->region);

fprintf(fileptr, "%d\n", temp->travelClass);

fprintf(fileptr, "%c\n", temp->tripCount);

fprintf(fileptr, "%c\n", temp->duration);

temp = temp->next;

} // while there is still passenger in the list

fclose(fileptr);

} // if file pointer is NULL

} // updateLinkedList()

// user login for accessing the program (return 1 if login success, 0 if fail)

int login() {

FILE\* fileptr;

char username[7];

char inputName[30];

char password[7];

char inputPass[30];

int i = 0;

char c;

printf("Username: ");

scanf("%s", inputName);

printf("Password: ");

// mask every character of the password input to '\*'

do {

c = getch();

if (c != '\r') {

if (c != '\b') {

// save the character and mask with '\*'

inputPass[i] = c;

printf("\*");

i++;

}

else {

// decrement 1 to the counter and delete the character when backspace key pressed

i--;

// if there is no character, set counter back to 0, else print '\b' to delete a '\*' from console

if (i < 0)

i++;

else

printf("\b \b");

} // if not backspace key

} // if not enter key

} while (c != '\r'); // do..while until user press enter key

inputPass[i] = '\0';

fileptr = fopen(LOGIN, "r");

if (fileptr == NULL) {

// output warning

printf("!- Warning -!\n");

printf("!- The File \"passenger.txt\" Does Not Exist -!\n");

printf("!- Warning -!\n");

}

else {

while (!feof(fileptr)) {

// get login data from "login.txt"

fscanf(fileptr, "Username: %s\n", username);

fscanf(fileptr, "Password: %s\n", password);

if (strcmp(inputName, username) == 0 && strcmp(inputPass, password) == 0) {

// useless loading console output (user experience)

printf("\nLogin in");

Sleep(250);

printf(".");

Sleep(250);

printf(".");

Sleep(250);

printf(".");

Sleep(250);

printf("\n");

fclose(fileptr);

return 1;

} // if input and login data match

} // while not end of file

fclose(fileptr);

return 0;

} // if file pointer is NULL

} // login()

// print all passengers' details and all criteria statistics into "report.txt"

void printReport(struct passenger\* top) {

FILE\* fileptr;

int finish;

struct passenger\* temp;

int passengerCount = 1;

char region[15];

char travelClass[16];

char tripCount[32];

char duration[18];

fileptr = fopen(REPORT, "w");

if (fileptr == NULL) {

// output warning

printf("!- Warning -!\n");

printf("!- The File \"report.txt\" Does Not Exist -!\n");

printf("!- Warning -!\n");

}

else {

temp = top;

// useless loading console output (user experience)

printf("\nWriting to \"report.txt\"");

Sleep(250);

printf(".");

Sleep(250);

printf(".");

Sleep(250);

printf(".");

Sleep(250);

printf("\n");

fprintf(fileptr, "===================Passenger Database===================");

while (temp != NULL) {

// print all passengers' details

fprintf(fileptr, "\n==Passenger %d==\n", passengerCount);

fprintf(fileptr, "Passport Number: %d\n", temp->passportNum);

fprintf(fileptr, "First Name: %s\n", temp->firstName);

fprintf(fileptr, "Second Name: %s\n", temp->secondName);

fprintf(fileptr, "Year Born: %d\n", temp->yearBorn);

fprintf(fileptr, "Email Address: %s\n", temp->email);

strcpy(region, getRegion(temp->region));

strcpy(travelClass, getTravelClass(temp->travelClass));

strcpy(tripCount, getTripCount(temp->tripCount));

strcpy(duration, getDuration(temp->duration));

fprintf(fileptr, "Region: %s\n", region);

fprintf(fileptr, "Travel Class: %s\n", travelClass);

fprintf(fileptr, "Trip Count to Ireland: %s\n", tripCount);

fprintf(fileptr, "Stay Duration: %s\n", duration);

fprintf(fileptr, "================\n");

temp = temp->next;

passengerCount++;

} // while iterate through the linked list

fclose(fileptr);

printStatistics(top);

printf("\nFile Writing Complete.\n");

} // if file pointer is NULL

} // printReport()

// print all criteria statistics into "report.txt" (append to the bottom of passenger details)

void printStatistics(struct passenger\* top) {

FILE\* fileptr;

fileptr = fopen(REPORT, "a");

if (fileptr == NULL) {

// output warning

printf("!- Warning -!\n");

printf("!- The File \"report.txt\" Does Not Exist -!\n");

printf("!- Warning -!\n");

}

else {

// print all criteria statistics

fprintf(fileptr, "\n==========Statistics From Criteria \"Economy\"==========\n");

calculateStatistics(1, top, 1, fileptr);

fprintf(fileptr, "\n==========Statistics From Criteria \"Premium Economy\"==========\n");

calculateStatistics(2, top, 1, fileptr);

fprintf(fileptr, "\n==========Statistics From Criteria \"Business Class\"==========\n");

calculateStatistics(3, top, 1, fileptr);

fprintf(fileptr, "\n==========Statistics From Criteria \"First Class\"==========\n");

calculateStatistics(4, top, 1, fileptr);

fprintf(fileptr, "\n==========Statistics From Criteria \"Born Before 1980\"==========\n");

calculateStatistics(5, top, 1, fileptr);

} // if file pointer is NULL

fclose(fileptr);

} // printStatistics()

## Text Files

### login.txt

Username: admin1

Password: admin1

Username: manage

Password: aaa111

Username: guest2

Password: 123456

### passenger.txt

18576019

Van

Damme

1960

expandables@lionheart.com

2

1

a

b

19401127

Bruce

Lee

1940

kung@fu.com

3

2

a

b

19460614

Donald

Trump

1946

donald\_trump@gmail.com

4

4

a

b

19540129

Oprah

Winfrey

1954

operah@hotmail.com

4

3

a

c

19550828

Bill

Gates

1955

microsoft@outlook.com

4

3

a

c

19681012

Hugh

Jackman

1968

wolverine@marvel.com

5

1

b

a

19710628

Elon

Musk

1971

tesla@spacex.com

2

4

a

a

19750502

David

Beckham

1975

beckham@live.com

1

2

c

d

19840514

Mark

Zuckerberg

1984

facebook@analytica.com

4

3

c

d

19850205

Cristiano

Ronaldo

1985

football4life@eksdee.com

2

4

b

a

19888552

Emma

Watson

1990

hermione@hogwarts.com

1

3

c

b

19970605

Bernard

Wong

1997

G00341962@gmit.com

3

1

b

d

23918475

Chris

Hemsworth

1983

thor@marvel.com

5

2

c

c

46850007

Daniel

Craig

1968

james@bond.com

1

1

c

d

54271963

Donnie

Yen

1963

ip@man.com

3

4

a

c

64195682

Jackie

Chan

1954

amazing@idol.com

3

2

c

a

71439208

Simon

Cowell

1959

britian@gotTalent.com

1

3

b

b

91827365

Daniel

Radcliffe

1989

potter@hogwarts.com

1

1

a

c

97865643

Cara

Delevingne

1992

cara@del.com

1

2

a

a

### report.txt

===================Passenger Database===================

==Passenger 1==

Passport Number: 18576019

First Name: Van

Second Name: Damme

Year Born: 1960

Email Address: expandables@lionheart.com

Region: Rest of Europe

Travel Class: Economy

Trip Count to Ireland: Less than three times per year

Stay Duration: Less than 3 days

================

==Passenger 2==

Passport Number: 19401127

First Name: Bruce

Second Name: Lee

Year Born: 1940

Email Address: kung@fu.com

Region: Asia

Travel Class: Premium Economy

Trip Count to Ireland: Less than three times per year

Stay Duration: Less than 3 days

================

==Passenger 3==

Passport Number: 19460614

First Name: Donald

Second Name: Trump

Year Born: 1946

Email Address: donald\_trump@gmail.com

Region: Americas

Travel Class: First Class

Trip Count to Ireland: Less than three times per year

Stay Duration: Less than 3 days

================

==Passenger 4==

Passport Number: 19540129

First Name: Oprah

Second Name: Winfrey

Year Born: 1954

Email Address: operah@hotmail.com

Region: Americas

Travel Class: Business Class

Trip Count to Ireland: Less than three times per year

Stay Duration: Less than 7 days

================

==Passenger 5==

Passport Number: 19550828

First Name: Bill

Second Name: Gates

Year Born: 1955

Email Address: microsoft@outlook.com

Region: Americas

Travel Class: Business Class

Trip Count to Ireland: Less than three times per year

Stay Duration: Less than 7 days

================

==Passenger 6==

Passport Number: 19681012

First Name: Hugh

Second Name: Jackman

Year Born: 1968

Email Address: wolverine@marvel.com

Region: Australasia

Travel Class: Economy

Trip Count to Ireland: Less than five times per year

Stay Duration: One day

================

==Passenger 7==

Passport Number: 19710628

First Name: Elon

Second Name: Musk

Year Born: 1971

Email Address: tesla@spacex.com

Region: Rest of Europe

Travel Class: First Class

Trip Count to Ireland: Less than three times per year

Stay Duration: One day

================

==Passenger 8==

Passport Number: 19750502

First Name: David

Second Name: Beckham

Year Born: 1975

Email Address: beckham@live.com

Region: UK

Travel Class: Premium Economy

Trip Count to Ireland: More than five times per year

Stay Duration: More than 7 days

================

==Passenger 9==

Passport Number: 19840514

First Name: Mark

Second Name: Zuckerberg

Year Born: 1984

Email Address: facebook@analytica.com

Region: Americas

Travel Class: Business Class

Trip Count to Ireland: More than five times per year

Stay Duration: More than 7 days

================

==Passenger 10==

Passport Number: 19850205

First Name: Cristiano

Second Name: Ronaldo

Year Born: 1985

Email Address: football4life@eksdee.com

Region: Rest of Europe

Travel Class: First Class

Trip Count to Ireland: Less than five times per year

Stay Duration: One day

================

==Passenger 11==

Passport Number: 19888552

First Name: Emma

Second Name: Watson

Year Born: 1990

Email Address: hermione@hogwarts.com

Region: UK

Travel Class: Business Class

Trip Count to Ireland: More than five times per year

Stay Duration: Less than 3 days

================

==Passenger 12==

Passport Number: 19970605

First Name: Bernard

Second Name: Wong

Year Born: 1997

Email Address: G00341962@gmit.com

Region: Asia

Travel Class: Economy

Trip Count to Ireland: Less than five times per year

Stay Duration: More than 7 days

================

==Passenger 13==

Passport Number: 23918475

First Name: Chris

Second Name: Hemsworth

Year Born: 1983

Email Address: thor@marvel.com

Region: Australasia

Travel Class: Premium Economy

Trip Count to Ireland: More than five times per year

Stay Duration: Less than 7 days

================

==Passenger 14==

Passport Number: 46850007

First Name: Daniel

Second Name: Craig

Year Born: 1968

Email Address: james@bond.com

Region: UK

Travel Class: Economy

Trip Count to Ireland: More than five times per year

Stay Duration: More than 7 days

================

==Passenger 15==

Passport Number: 54271963

First Name: Donnie

Second Name: Yen

Year Born: 1963

Email Address: ip@man.com

Region: Asia

Travel Class: First Class

Trip Count to Ireland: Less than three times per year

Stay Duration: Less than 7 days

================

==Passenger 16==

Passport Number: 64195682

First Name: Jackie

Second Name: Chan

Year Born: 1954

Email Address: amazing@idol.com

Region: Asia

Travel Class: Premium Economy

Trip Count to Ireland: More than five times per year

Stay Duration: One day

================

==Passenger 17==

Passport Number: 71439208

First Name: Simon

Second Name: Cowell

Year Born: 1959

Email Address: britian@gotTalent.com

Region: UK

Travel Class: Business Class

Trip Count to Ireland: Less than five times per year

Stay Duration: Less than 3 days

================

==Passenger 18==

Passport Number: 91827365

First Name: Daniel

Second Name: Radcliffe

Year Born: 1989

Email Address: potter@hogwarts.com

Region: UK

Travel Class: Economy

Trip Count to Ireland: Less than three times per year

Stay Duration: Less than 7 days

================

==Passenger 19==

Passport Number: 97865643

First Name: Cara

Second Name: Delevingne

Year Born: 1992

Email Address: cara@del.com

Region: UK

Travel Class: Premium Economy

Trip Count to Ireland: Less than three times per year

Stay Duration: One day

================

==========Statistics From Criteria "Economy"==========

Percentage of passengers who travel from the UK : 40.00%

Percentage of passengers who travel from the Rest of Europe : 20.00%

Percentage of passengers who travel from the Asia : 20.00%

Percentage of passengers who travel from the Americas : 0.00%

Percentage of passengers who travel from the Australasia : 20.00%

Percentage of passengers who spent on average one day in Ireland : 20.00%

Percentage of passengers who spent on average less than 3 days in Ireland : 20.00%

Percentage of passengers who spent on average less than 7 days in Ireland : 20.00%

Percentage of passengers who spent on average more than 7 days in Ireland : 40.00%

==========Statistics From Criteria "Premium Economy"==========

Percentage of passengers who travel from the UK : 40.00%

Percentage of passengers who travel from the Rest of Europe : 0.00%

Percentage of passengers who travel from the Asia : 40.00%

Percentage of passengers who travel from the Americas : 0.00%

Percentage of passengers who travel from the Australasia : 20.00%

Percentage of passengers who spent on average one day in Ireland : 40.00%

Percentage of passengers who spent on average less than 3 days in Ireland : 20.00%

Percentage of passengers who spent on average less than 7 days in Ireland : 20.00%

Percentage of passengers who spent on average more than 7 days in Ireland : 20.00%

==========Statistics From Criteria "Business Class"==========

Percentage of passengers who travel from the UK : 40.00%

Percentage of passengers who travel from the Rest of Europe : 0.00%

Percentage of passengers who travel from the Asia : 0.00%

Percentage of passengers who travel from the Americas : 60.00%

Percentage of passengers who travel from the Australasia : 0.00%

Percentage of passengers who spent on average one day in Ireland : 0.00%

Percentage of passengers who spent on average less than 3 days in Ireland : 40.00%

Percentage of passengers who spent on average less than 7 days in Ireland : 40.00%

Percentage of passengers who spent on average more than 7 days in Ireland : 20.00%

==========Statistics From Criteria "First Class"==========

Percentage of passengers who travel from the UK : 0.00%

Percentage of passengers who travel from the Rest of Europe : 50.00%

Percentage of passengers who travel from the Asia : 25.00%

Percentage of passengers who travel from the Americas : 25.00%

Percentage of passengers who travel from the Australasia : 0.00%

Percentage of passengers who spent on average one day in Ireland : 50.00%

Percentage of passengers who spent on average less than 3 days in Ireland : 25.00%

Percentage of passengers who spent on average less than 7 days in Ireland : 25.00%

Percentage of passengers who spent on average more than 7 days in Ireland : 0.00%

==========Statistics From Criteria "Born Before 1980"==========

Percentage of passengers who travel from the UK : 25.00%

Percentage of passengers who travel from the Rest of Europe : 16.67%

Percentage of passengers who travel from the Asia : 25.00%

Percentage of passengers who travel from the Americas : 25.00%

Percentage of passengers who travel from the Australasia : 8.33%

Percentage of passengers who spent on average one day in Ireland : 25.00%

Percentage of passengers who spent on average less than 3 days in Ireland : 33.33%

Percentage of passengers who spent on average less than 7 days in Ireland : 25.00%

Percentage of passengers who spent on average more than 7 days in Ireland : 16.67%

# Screenshots

