**Object Orientated Programming Coursework – COA256 – Plane Route Planner**

**Designs**

My program consists of the main program, 5 structs and 9 methods. Many methods where originally written in the main program, but I lifted them out. This was either because they were used more than once in the program or the fact that it tidied my program up.

Structs

I have 5 structs which can be organised into two groups. The first are the airport and flight structs. They both are used when the program is loaded to fill vectors with all the information in the text files.

Then there are the dirFlight, conFlight and twoConFlight structs. These are used to hold information about possible journeys each of which correspond to; Direct journeys, journeys with one connection and journeys with two connections. I set these up as it was a lot easier and less complicated to lift information from the airport and flight vectors and put it into these structs earlier on as opposed to accessing the vectors later and trying to correlate it to a journey.

Methods

getMonth – this simply takes the integer selected as the month and returns the name of the month eg. Input = 2, output = February

getAirports – this streams in the information from the airportinfo.txt file and creates airport objects. It then puts these objects into a vector which it returns.

getFights – this streams in the information from the flights.txt file and creates flight objects. It then puts these objects into a vector which it returns.

printDirFlights – this takes a vector of dirFlight objects and prints out all the information onto the screen.

timeStr – this take an integer, an amount on minutes, and turns it into a string with the format xx hour/hours and xx minute/minutes, depending on the value of the int.

*printConFlights and printTwoConFlights do the same but with conFlight and twoConFlight objects.*

bookDirFlight – this takes a vector of dirFlight objects, values for the date of booking and the chosen flight number and makes a booking. It finds the selected journey and prints all the information to the screen and also saves it to a receipt, named receipt.txt

*bookConFlights and bookTwoConFlights do the same but with conFlight and twoConFlight objects.*

Main Program

Firstly it calls the getAirport and getFlight methods to load in all the information from the text files. It then prints out a list of all the airports that the system works with.

It then calls for the first user input, an airport name or code which the user wishes to depart from. It checks for errors and then continues to ask for preferred arrival airport.

It then goes on to ask for the day, month and year that the user wishes to travel, whilst checking each one for errors and turning to integer month into a string with the name of the month chosen.

It then searches using a for loop for direct journeys and if it finds any, it turns the flight information, along with both airport taxes into a dirFlight object which is then stored in a vector. If this vector is empty after the search it then goes on to search for connecting flights, using two for loops. If it finds a connecting flight, similarly to before, it turns the flight information, along with all airport tax into a conFlight object, and stores it in an array. Again if no connecting flights are found it then searches for flights with two connections, using three for loops. Again it turns all the information into an object and stores it in an vector.

The results are then printed, and sorting options are given. The sorting options are by cost and by journey time. If it’s dealing with direct flights a specific airline sort is also offered. If cost or journey sorts are called, the program sorts them by using a simple bubble sort. If a specific airline sort is chosen, it simply searches through the vector of results and deletes any entry which does not correspond. If a sort is chosen then the results are reprinted.

The program then asks the user which flight they would like to book by choosing a corresponding number. When a valid number is chosen it then outputs all the information about the journey, by calling the related bookFlight method, and saves the information to a receipt. The program then ends.

**Functionality working**

The user can successfully enter a start and end airport using the airport name or code, to travel from and to.

The program will then search through the flights and return all the direct flights. If no direct flights are found the program will search for and return all the flights with one connection. If there are no flights with one connection, the program will then search for and return flights with two connections. If none of these are found the program simply tells the user there are no flights available.

Once a list of available flights is returned, the user can choose to order them by cost or journey time or continue to book the flights. If direct flights have been returned, the user can choose to only look at flights using one airline.

Once a new sorted list of flights has been produced or the user clicks continue, the user can then choose a flight to book. Once chosen it outputs all the information about the journey onto the screen and also into a text file called receipt.

**Functionality not completed**

The date has to be entered in three separate parts; day, month, year as I found it hard to distinguish between the three when they were entered in 1 string; day/month/year.

**Test plans**

Entering a airport

Enter: Birmingham Result: Birmingham is used

Enter: BHX Result: Birmingham is used

Enter: Birmingh Result: Error message produced

Entering a day

Enter: 1 Result: 1 is used as the day

Enter 31 Result: Error message is produced

Entering a month

Enter: 4 Result: 1 is used as the month

Enter 12 Result: Error message is produced

Entering a year

Enter: 2010 Result: 2010 is used as the year

Enter: 2021 Result: An error message is produced

Choosing Options

*The same method is used for most of the program in the below examples the amount of options is 4*

Enter: 1 Result: The option 1 is used

Enter: 5 Result: An error message is produced

Testing a Direct Flight

Input: BHX to PAR

Returns: 3 Direct Flights

Testing a flight with one Connection

Input: BHX to MLN

Returns: 7 possible journeys all with one connection

Testing a flight with two Connections

Input: LHR to VIE

Returns: 27 possible journeys all with two connections

Testing a flight with valid airports but no possible journey

Input: Man to VIE

Returns: “We are very sorry. There are no possible journeys between the cities you have chosen”

**Class diagrams**

main()

flight

Attributes

string airline, depart, arrive

int time, cost

airport

Attributes

string name, code

int tax, connectionTime

dirFlight

Attributes

string airline, depart, arrive

int time, cost, tax1, tax2;

conFlight

Attributes

string airline1, airline2, depart1, arrive1, depart2, arrive2

int time1, time2, conTime, cost1, cost2, tax1, tax2, tax3

twoConFlight

Attributes

string airline1, airline2, airline3, depart1, arrive1, depart2, arrive2, depart3, arrive3

int time1, time2, time3, conTime1, conTime2, cost1, cost2, cost3, tax1, tax2, tax3, tax4

Operations

string getMonth(int a)

vector<airport> getAirports()

vector<flight> getFlights()

int timeStr(int a)

void printDirFlights()

void printConFlights()

void printTwoConFlights()

void bookDirFlight()

void bookConFlight()

void bookTwoConFlight()

**Program Listings**

#include "std\_lib\_facilities.h"

struct airport

{

string name, code;

int tax, connectionTime;

airport(string a, string b, int c, int d)

:name(a), code(b), tax(c), connectionTime(d) {}

};

struct flight

{

string airline, depart, arrive;

int time, cost;

flight(string a, string b, string c, int d, int e)

:airline(a), depart(b), arrive(c), time(d), cost(e) {}

};

struct dirFlight //direct flights

{

string airline, depart, arrive;

int time, cost, tax1, tax2;

dirFlight(string a, string b, string c, int d, int e, int f, int g)

:airline(a), depart(b), arrive(c), time(d), cost(e), tax1(f), tax2(g) {}

};

struct conFlight //flights with 1 connection

{

string airline1, airline2, depart1, arrive1, depart2, arrive2;

int time1, time2, conTime, cost1, cost2, tax1, tax2, tax3;

conFlight(string a, string b, string c, string d, string e, string f, int g, int h, int i, int j, int l, int m, int n, int o)

:airline1(a), airline2(b), depart1(c), arrive1(d), depart2(e), arrive2(f), time1(g), time2(h), conTime(i), cost1(j), cost2(l), tax1(m), tax2(n), tax3(o) {}

};

struct twoConFlight //flights with two connections

{

string airline1, airline2, airline3, depart1, arrive1, depart2, arrive2, depart3, arrive3;

int time1, time2, time3, conTime1, conTime2, cost1, cost2, cost3, tax1, tax2, tax3, tax4;

twoConFlight(string a, string b, string c, string d, string e, string f, string g, string h, string i, int j, int k, int l, int m, int n, int o, int p, int q, int r, int s, int t, int u)

:airline1(a), airline2(b), airline3(c), depart1(d), arrive1(e), depart2(f), arrive2(g), depart3(h), arrive3(i), time1(j), time2(k), time3(l), conTime1(m), conTime2(n), cost1(o), cost2(p), cost3(q), tax1(r), tax2(s), tax3(t), tax4(u) {}

};

string getMonth(int a){

string monthString;

switch (a) {

case 1:

monthString = "January";

break;

case 2:

monthString = "February";

break;

case 3:

monthString = "March";

break;

case 4:

monthString = "April";

break;

case 5:

monthString = "May";

break;

case 6:

monthString = "June";

break;

case 7:

monthString = "July";

break;

case 8:

monthString = "August";

break;

case 9:

monthString = "September";

break;

case 10:

monthString = "October";

break;

case 11:

monthString = "November";

break;

case 12:

monthString = "December";

break;

}

return monthString;

}

vector<airport> getAirports() //makes a vector of the airports

{

ifstream istAirport("airportinfo.txt");

vector<airport>airportInfo;

string istName, istCode;

int istTax, istConnectionTime;

while (istAirport >> istName >> istCode >> istTax >> istConnectionTime){

airportInfo.push\_back(airport(istName,istCode,istTax,istConnectionTime));}

return airportInfo;

}

vector<flight> getFlights() //makes a vector of the flights

{

ifstream istFlights("flights.txt");

vector<flight>flightInfo;

string istAirline, istDepart, istArrive, airportList;

int istTime, istCost;

while (istFlights >> istAirline >> istDepart >> istArrive >> istTime >> istCost){

flightInfo.push\_back(flight(istAirline,istDepart,istArrive,istTime,istCost));}

return flightInfo;

}

string timeStr(int a){

if (a>=60){

int hours = a/60;

int remMin = a-(hours\*60);

string strHours;

stringstream out;

out << hours;

strHours = out.str();

string strMins;

stringstream out2;

out2 << remMin;

strMins = out2.str();

if (hours>1){

if (remMin>1){return strHours+" hours and "+strMins+" minutes";}

if (remMin==1){return strHours+" hours and "+strMins+" minute";}

if (remMin==0){return strHours+" hours";}}

if (hours==1){

if (remMin>1){return strHours+" hour and "+strMins+" minutes";}

if (remMin==1){return strHours+" hour and "+strMins+" minute";}

if (remMin==0){return strHours+" hour";}}}

if (a<60){

string strMins;

stringstream out;

out << a;

strMins = out.str();

return strMins+" minutes";}}

void printDirFlights(vector<dirFlight>dirFlights){

int noOptions = 0;

char pound=156;

for(int x=0; x<dirFlights.size(); x++){

string possAirline = dirFlights[x].airline;

int possTime = dirFlights[x].time;

double possCost = (dirFlights[x].cost\*(dirFlights[x].tax1)/100)+(dirFlights[x].cost\*(dirFlights[x].tax2)/100) + dirFlights[x].cost;

noOptions++;

cout <<"\n"<<noOptions<< ". Direct Flight - Total Journey Time: "+timeStr(possTime)+" - Airline: "+possAirline+" - Cost (inc tax): "<<pound<<possCost<<"\n";}}

void printConFlights(vector<conFlight>conFlights){

int noOptions = 0;

char pound=156;

for(int x=0; x<conFlights.size(); x++){

string possAirline;

if(conFlights[x].airline1 == conFlights[x].airline2){possAirline = conFlights[x].airline1;}else{possAirline="Multiple";};

int possTime = conFlights[x].time1 + conFlights[x].time2 + conFlights[x].conTime;

double possCost = (conFlights[x].cost1\*(conFlights[x].tax1)/100)+(conFlights[x].cost1\*(conFlights[x].tax2)/100)+(conFlights[x].cost2\*(conFlights[x].tax2)/100)+(conFlights[x].cost2\*(conFlights[x].tax3)/100) + conFlights[x].cost1 + conFlights[x].cost2;

noOptions++;

cout <<"\n"<<noOptions<< ". Number of Connections: 1 - Total Journey Time: "+timeStr(possTime)+" - Airline: "+possAirline+" - Cost (inc tax): "<<pound<<possCost<<"\n";}}

void printTwoConFlights(vector<twoConFlight>twoConFlights){

int noOptions = 0;

char pound=156;

for(int x=0; x<twoConFlights.size(); x++){

string possAirline;

if(twoConFlights[x].airline1 == twoConFlights[x].airline2 && twoConFlights[x].airline1 == twoConFlights[x].airline3){possAirline = twoConFlights[x].airline1;}else{possAirline = "Multiple";}

int possTime = twoConFlights[x].time1 + twoConFlights[x].time2 + twoConFlights[x].time3 + twoConFlights[x].conTime1 + twoConFlights[x].conTime2;

double possCost = (twoConFlights[x].cost1\*(twoConFlights[x].tax1)/100)+(twoConFlights[x].cost1\*(twoConFlights[x].tax2)/100)+(twoConFlights[x].cost2\*(twoConFlights[x].tax2)/100)+(twoConFlights[x].cost2\*(twoConFlights[x].tax3)/100)+(twoConFlights[x].cost3\*(twoConFlights[x].tax3)/100)+(twoConFlights[x].cost3\*(twoConFlights[x].tax4)/100)+twoConFlights[x].cost1+twoConFlights[x].cost2+twoConFlights[x].cost3;

noOptions++;

cout <<"\n"<<noOptions<< ". Number of Connections: 2 - Total Journey Time: "+timeStr(possTime)+" - Airline: "+possAirline+" - Cost (inc tax): "<<pound<<possCost<<"\n";}}

void bookDirFlight(vector<dirFlight>dirFlights, string monthStr, int day, int year, int flightNo){

char pound=156;

ofstream receipt("receipt.txt");

double initialCost = dirFlights[flightNo].cost;

double tax1 = dirFlights[flightNo].tax1;

double tax2 = dirFlights[flightNo].tax2;

double totalTax = ((tax1/100)\*initialCost)+((tax2/100)\*initialCost);

double finalCost = initialCost+totalTax;

cout << "\n \n Flight Booking Information: \n Date of travel: " <<day<<" "+monthStr+" "<<year<< "\n Depart: "+dirFlights[flightNo].depart+"\n Arrive: "+dirFlights[flightNo].arrive+"\n Flight Time: "+timeStr(dirFlights[flightNo].time)+"\n Airline: "+dirFlights[flightNo].airline+"\n Departure airport tax: "<<dirFlights[flightNo].tax1<<"% Arrival airport tax: "<<dirFlights[flightNo].tax2<<"%\n Flight Cost: "<<pound<<initialCost<<"\n Total Tax: "<<pound<<totalTax<<"\n Total Flight Cost: "<<pound<<finalCost<<"\n \n";

receipt << "\n \n Flight Booking Information: \n Date of travel: " <<day<<" "+monthStr+" "<<year<< "\n Depart: "+dirFlights[flightNo].depart+"\n Arrive: "+dirFlights[flightNo].arrive+"\n Flight Time: "+timeStr(dirFlights[flightNo].time)+"\n Airline: "+dirFlights[flightNo].airline+"\n Departure airport tax: "<<dirFlights[flightNo].tax1<<"% Arrival airport tax: "<<dirFlights[flightNo].tax2<<"%\n Flight Cost: £"<<initialCost<<"\n Total Tax: £"<<totalTax<<"\n Total Flight Cost: £"<<finalCost<<"\n \n";

receipt.close();

}

void bookConFlight(vector<conFlight>conFlights, string monthStr, int day, int year, int flightNo){

char pound=156;

ofstream receipt("receipt.txt");

double f1initialCost = conFlights[flightNo].cost1;

double f1tax1 = conFlights[flightNo].tax1;

double f1tax2 = conFlights[flightNo].tax2;

double f1totalTax = ((f1tax1/100)\*f1initialCost)+((f1tax2/100)\*f1initialCost);

double f1finalCost = f1initialCost+f1totalTax;

cout << "\n Flight Booking Information: \n Date of travel: "<<day<<" "+monthStr+" "<<year;

cout << "\n \n Flight 1: \n Depart: "+conFlights[flightNo].depart1+"\n Arrive: "+conFlights[flightNo].arrive1+"\n Flight Time: "+timeStr(conFlights[flightNo].time1)+"\n Airline: "+conFlights[flightNo].airline1+"\n Departure airport tax: "<<conFlights[flightNo].tax1<<"% Arrival airport tax: "<<conFlights[flightNo].tax2<<"%\n Flight Cost: "<<pound<<f1initialCost<<"\n Total Tax: "<<pound<<f1totalTax<<"\n Total Flight Cost: "<<pound<<f1finalCost<<"\n";

receipt << "\n Flight Booking Information: \n Date of travel: "<<day<<" "+monthStr+" "<<year;

receipt << "\n \n Flight 1: \n Depart: "+conFlights[flightNo].depart1+"\n Arrive: "+conFlights[flightNo].arrive1+"\n Flight Time: "+timeStr(conFlights[flightNo].time1)+"\n Airline: "+conFlights[flightNo].airline1+"\n Departure airport tax: "<<conFlights[flightNo].tax1<<"% Arrival airport tax: "<<conFlights[flightNo].tax2<<"%\n Flight Cost: £"<<f1initialCost<<"\n Total Tax: £"<<f1totalTax<<"\n Total Flight Cost: £"<<f1finalCost<<"\n";

cout<<"\n Connection time in "+conFlights[flightNo].arrive1+": "+timeStr(conFlights[flightNo].conTime)+"\n";

receipt << "\n Connection time in "+conFlights[flightNo].arrive1+": "+timeStr(conFlights[flightNo].conTime)+"\n";

double f2initialCost = conFlights[flightNo].cost2;

double f2tax1 = conFlights[flightNo].tax2;

double f2tax2 = conFlights[flightNo].tax3;

double f2totalTax = ((f2tax1/100)\*f2initialCost)+((f2tax2/100)\*f2initialCost);

double f2finalCost = f2initialCost+f2totalTax;

cout << "\n Flight 2: \n Depart: "+conFlights[flightNo].depart2+"\n Arrive: "+conFlights[flightNo].arrive2+"\n Flight Time: "+timeStr(conFlights[flightNo].time2)+"\n Airline: "+conFlights[flightNo].airline2+"\n Departure airport tax: "<<conFlights[flightNo].tax2<<"% Arrival airport tax: "<<conFlights[flightNo].tax3<<"%\n Flight Cost: "<<pound<<f2initialCost<<"\n Total Tax: "<<pound<<f2totalTax<<"\n Total Flight Cost: "<<pound<<f2finalCost<<"\n";

receipt << "\n Flight 2: \n Depart: "+conFlights[flightNo].depart2+"\n Arrive: "+conFlights[flightNo].arrive2+"\n Flight Time: "+timeStr(conFlights[flightNo].time2)+"\n Airline: "+conFlights[flightNo].airline2+"\n Departure airport tax: "<<conFlights[flightNo].tax2<<"% Arrival airport tax: "<<conFlights[flightNo].tax3<<"%\n Flight Cost: £"<<f2initialCost<<"\n Total Tax: £"<<f2totalTax<<"\n Total Flight Cost: £"<<f2finalCost<<"\n";

double totalCost = f1finalCost + f2finalCost;

int totalTime = conFlights[flightNo].time1 + conFlights[flightNo].time2 + conFlights[flightNo].conTime;

cout<< "\n Total Cost: "<<pound<<totalCost<<"\n Total Time: "+timeStr(totalTime)+"\n \n";

receipt << "\n Total Cost: £"<<totalCost<<"\n Total Time: "+timeStr(totalTime)+"\n \n";

receipt.close();

}

void bookTwoConFlight(vector<twoConFlight>twoConFlights, string monthStr, int day, int year, int flightNo){

char pound=156;

ofstream receipt("receipt.txt");

double f1initialCost = twoConFlights[flightNo].cost1;

double f1tax1 = twoConFlights[flightNo].tax1;

double f1tax2 = twoConFlights[flightNo].tax2;

double f1totalTax = ((f1tax1/100)\*f1initialCost)+((f1tax2/100)\*f1initialCost);

double f1finalCost = f1initialCost+f1totalTax;

cout << "\n Flight Booking Information: \n Date of travel: "<<day<<" "+monthStr+" "<<year;

cout << "\n \n Flight 1: \n Depart: "+twoConFlights[flightNo].depart1+"\n Arrive: "+twoConFlights[flightNo].arrive1+"\n Flight Time: "+timeStr(twoConFlights[flightNo].time1)+"\n Airline: "+twoConFlights[flightNo].airline1+"\n Departure airport tax: "<<twoConFlights[flightNo].tax1<<"% Arrival airport tax: "<<twoConFlights[flightNo].tax2<<"%\n Flight Cost: "<<pound<<f1initialCost<<"\n Total Tax: "<<pound<<f1totalTax<<"\n Total Flight Cost: "<<pound<<f1finalCost<<"\n";

receipt << "\n Flight Booking Information: \n Date of travel: "<<day<<" "+monthStr+" "<<year;

receipt << "\n \n Flight 1: \n Depart: "+twoConFlights[flightNo].depart1+"\n Arrive: "+twoConFlights[flightNo].arrive1+"\n Flight Time: "+timeStr(twoConFlights[flightNo].time1)+"\n Airline: "+twoConFlights[flightNo].airline1+"\n Departure airport tax: "<<twoConFlights[flightNo].tax1<<"% Arrival airport tax: "<<twoConFlights[flightNo].tax2<<"%\n Flight Cost: £"<<f1initialCost<<"\n Total Tax: £"<<f1totalTax<<"\n Total Flight Cost: £"<<f1finalCost<<"\n";

cout << "\n Connection time in "+twoConFlights[flightNo].arrive1+": "+timeStr(twoConFlights[flightNo].conTime1)+"\n";

receipt << "\n Connection time in "+twoConFlights[flightNo].arrive1+": "+timeStr(twoConFlights[flightNo].conTime1)+"\n";

double f2initialCost = twoConFlights[flightNo].cost2;

double f2tax1 = twoConFlights[flightNo].tax2;

double f2tax2 = twoConFlights[flightNo].tax3;

double f2totalTax = ((f2tax1/100)\*f2initialCost)+((f2tax2/100)\*f2initialCost);

double f2finalCost = f2initialCost+f2totalTax;

cout << "\n Flight 2: \n Depart: "+twoConFlights[flightNo].depart2+"\n Arrive: "+twoConFlights[flightNo].arrive2+"\n Flight Time: "+timeStr(twoConFlights[flightNo].time2)+"\n Airline: "+twoConFlights[flightNo].airline2+"\n Departure airport tax: "<<twoConFlights[flightNo].tax2<<"% Arrival airport tax: "<<twoConFlights[flightNo].tax3<<"%\n Flight Cost: "<<pound<<f2initialCost<<"\n Total Tax: "<<pound<<f2totalTax<<"\n Total Flight Cost: "<<pound<<f2finalCost<<"\n";

receipt << "\n Flight 2: \n Depart: "+twoConFlights[flightNo].depart2+"\n Arrive: "+twoConFlights[flightNo].arrive2+"\n Flight Time: "+timeStr(twoConFlights[flightNo].time2)+"\n Airline: "+twoConFlights[flightNo].airline2+"\n Departure airport tax: "<<twoConFlights[flightNo].tax2<<"% Arrival airport tax: "<<twoConFlights[flightNo].tax3<<"%\n Flight Cost: £"<<f2initialCost<<"\n Total Tax: £"<<f2totalTax<<"\n Total Flight Cost: £"<<f2finalCost<<"\n";

cout<<"\n Connection time in "+twoConFlights[flightNo].arrive2+": "+timeStr(twoConFlights[flightNo].conTime2)+"\n";

receipt <<"\n Connection time in "+twoConFlights[flightNo].arrive2+": "+timeStr(twoConFlights[flightNo].conTime2)+"\n";

double f3initialCost = twoConFlights[flightNo].cost3;

double f3tax1 = twoConFlights[flightNo].tax3;

double f3tax2 = twoConFlights[flightNo].tax4;

double f3totalTax = ((f3tax1/100)\*f3initialCost)+((f3tax2/100)\*f3initialCost);

double f3finalCost = f3initialCost+f3totalTax;

cout << "\n Flight 3: \n Depart: "+twoConFlights[flightNo].depart3+"\n Arrive: "+twoConFlights[flightNo].arrive3+"\n Flight Time: "+timeStr(twoConFlights[flightNo].time3)+"\n Airline: "+twoConFlights[flightNo].airline3+"\n Departure airport tax: "<<twoConFlights[flightNo].tax3<<"% Arrival airport tax: "<<twoConFlights[flightNo].tax4<<"%\n Flight Cost: "<<pound<<f3initialCost<<"\n Total Tax: "<<pound<<f3totalTax<<"\n Total Flight Cost: "<<pound<<f3finalCost<<"\n";

receipt << "\n Flight 3: \n Depart: "+twoConFlights[flightNo].depart3+"\n Arrive: "+twoConFlights[flightNo].arrive3+"\n Flight Time: "+timeStr(twoConFlights[flightNo].time3)+"\n Airline: "+twoConFlights[flightNo].airline3+"\n Departure airport tax: "<<twoConFlights[flightNo].tax3<<"% Arrival airport tax: "<<twoConFlights[flightNo].tax4<<"%\n Flight Cost: £"<<f3initialCost<<"\n Total Tax: £"<<f3totalTax<<"\n Total Flight Cost: £"<<f3finalCost<<"\n";

double totalCost = f1finalCost + f2finalCost + f3finalCost;

int totalTime = twoConFlights[flightNo].time1 + twoConFlights[flightNo].time2 + twoConFlights[flightNo].time3 + twoConFlights[flightNo].conTime1 + twoConFlights[flightNo].conTime2;

cout<< "\n Total Cost: "<<pound<<totalCost<<"\n Total Time: "+timeStr(totalTime)+"\n \n";

receipt << "\n Total Cost: £"<<totalCost<<"\n Total Time: "+timeStr(totalTime)+"\n \n";

receipt.close();

}

int main(){

vector<airport>airportInfo = getAirports();

int noAirports = airportInfo.size();

vector<flight>flightInfo = getFlights();

int noFlights = flightInfo.size();

cout << "\nHere is a list of all the airports that our flight planner system works with, and their airport codes:\n";

string airportList;

for(int x = 0; x<noAirports; x++){

airportList = airportList+airportInfo[x].name+"("+airportInfo[x].code+")\n";}

cout << "\n"+airportList+"\n";

int airportDepartNo = -1;

try {cout << "\nWhere do you want to fly from? \n";

string chosenDepart;

cin >> chosenDepart;

for(int x=0; x<noAirports; x++){ //checking if the user has entered an airport name or code

if(chosenDepart == airportInfo[x].name || chosenDepart == airportInfo[x].code){

airportDepartNo = x;}}

if(airportDepartNo==-1){error("You did not enter a valid airport name or airport code, if entering a name please make sure you use a capital letter at the start.");}}

catch (runtime\_error& e){

cerr << "\n Invalid entry: " << e.what() << endl;

keep\_window\_open("q");

return 1;}

int airportArriveNo = -1;

try {cout << "\nWhere do you want to fly to? \n";

string chosenArrive;

cin >> chosenArrive;

for(int x=0; x<noAirports; x++){

if(chosenArrive == airportInfo[x].name || chosenArrive == airportInfo[x].code){

airportArriveNo = x;}}

if(airportArriveNo==-1){error("You did not enter a valid airport name or airport code, if entering a name please make sure you use a capital letter at the start.");}}

catch (runtime\_error& e){

cerr << "\n Invalid entry: " << e.what() << endl;

keep\_window\_open("q");

return 1;}

int day,month,year;

try {cout << "\nWhat day do you want to fly on? (1-30) \n";

cin >> day;

if(day>30 || day<1){error("You need to enter a day between 1 and 30 \n");}}

catch (runtime\_error& e){

cerr << "\n Invalid entry: " << e.what() << endl;

keep\_window\_open("q");

return 1;}

try {cout << "\nWhat month do you want to fly on? (1-12) \n";

cin >> month;

if(month>12 || month<1){error("You need to enter a month between 1 and 12 \n");}}

catch (runtime\_error& e){

cerr << "\n Invalid entry: " << e.what() << endl;

keep\_window\_open("q");

return 1;}

try {cout << "\nWhat year do you want to fly on? \n";

cin >> year;

if(2009>=year || year>=2021){error("You need to enter a year between 2010 and 2020 \n");}}

catch (runtime\_error& e){

cerr << "\n Invalid entry: " << e.what() << endl;

keep\_window\_open("q");

return 1;}

string monthStr = getMonth(month);

vector <dirFlight>dirFlights;

vector<conFlight>conFlights;

vector<twoConFlight>twoConFlights;

for(int x=0; x<noFlights; x++){ //for loop searching for direct flights

if(airportInfo[airportDepartNo].name == flightInfo[x].depart && airportInfo[airportArriveNo].name == flightInfo[x].arrive){

dirFlight flight(flightInfo[x].airline, flightInfo[x].depart, flightInfo[x].arrive, flightInfo[x].time, flightInfo[x].cost, airportInfo[airportDepartNo].tax, airportInfo[airportArriveNo].tax);

dirFlights.push\_back(flight);}}

if(dirFlights.size()==0){

for(int x=0; x<noFlights; x++){ //for loop searching for flights with one connection if no direct flights are found.

if(airportInfo[airportDepartNo].name == flightInfo[x].depart){

for(int y=0; y<noFlights; y++){

if(flightInfo[x].arrive==flightInfo[y].depart){

if(flightInfo[y].arrive==airportInfo[airportArriveNo].name){

int connectAirportNo = 0;

for(int a=0; a<noAirports; a++){

if(flightInfo[y].depart == airportInfo[a].name){

int connectAirportNo = a;}}

conFlight flights(flightInfo[x].airline, flightInfo[y].airline, flightInfo[x].depart, flightInfo[x].arrive, flightInfo[y].depart, flightInfo[y].arrive, flightInfo[x].time, flightInfo[y].time, airportInfo[connectAirportNo].connectionTime, flightInfo[x].cost, flightInfo[y].cost, airportInfo[airportDepartNo].tax, airportInfo[connectAirportNo].tax, airportInfo[airportArriveNo].tax);

conFlights.push\_back(flights);}}}}}}

if(dirFlights.size()==0 && conFlights.size()==0){

for(int x=0; x<noFlights; x++){ //for loop searching for flights with two connections if no direct flights or flights with one connection are found.

if(airportInfo[airportDepartNo].name == flightInfo[x].depart){

for(int y=0; y<noFlights; y++){

if(flightInfo[x].arrive==flightInfo[y].depart){

for(int z=0; z<noFlights; z++){

if(flightInfo[y].arrive==flightInfo[z].depart){

if(flightInfo[z].arrive==airportInfo[airportArriveNo].name){

int connectAirportNo1 = 0;

int connectAirportNo2 = 0;

for(int a=0; a<noAirports; a++){

if(flightInfo[y].depart == airportInfo[a].name){

int connectAirportNo1 = a;}

if(flightInfo[z].depart == airportInfo[a].name){

int connectAirportNo2 = a;}}

twoConFlight flights(flightInfo[x].airline, flightInfo[y].airline, flightInfo[z].airline, flightInfo[x].depart, flightInfo[x].arrive, flightInfo[y].depart, flightInfo[y].arrive, flightInfo[z].depart, flightInfo[z].arrive, flightInfo[x].time, flightInfo[y].time, flightInfo[z].time, airportInfo[connectAirportNo1].connectionTime, airportInfo[connectAirportNo2].connectionTime, flightInfo[x].cost, flightInfo[y].cost, flightInfo[z].cost, airportInfo[airportDepartNo].tax, airportInfo[connectAirportNo1].tax, airportInfo[connectAirportNo2].tax, airportInfo[airportArriveNo].tax);

twoConFlights.push\_back(flights);}}}}}}}}

if(dirFlights.size()==0 && conFlights.size()== 0 && twoConFlights.size() == 0){cout << "\nWe are very sorry. There are no possible journeys between the cities you have chosen \n"; keep\_window\_open("q");return 1;}

cout << "\nHere are your possible flight options for the journey "<<airportInfo[airportDepartNo].name<<" to "<<airportInfo[airportArriveNo].name<<":\n";

int noOptions;

//outputting the search results depending on which array was filled

if(dirFlights.size() != 0){

printDirFlights(dirFlights);}

if(conFlights.size() != 0){

printConFlights(conFlights);}

if(twoConFlights.size() != 0){

printTwoConFlights(twoConFlights);}

if(dirFlights.size() > 1 || conFlights.size() > 1 || twoConFlights.size() > 1 ){

int sorter;

try {int check;

if(dirFlights.size() != 0){cout << "\nWhat would you like to do now (choose a number and press enter): \n 1. Sort list by Journey Time \n 2. Sort list by Cost \n 3. Only see results from one airline \n 4. Continue to Booking \n"; check=4;}

else{cout << "\nWhat would you like to do now (choose a number and press enter): \n 1. Sort list by Journey Time \n 2. Sort list by Cost \n 3. Continue to Booking \n "; check=3;}

cin >> sorter;

if(sorter>check){error("You chose an invalid option");}}

catch (runtime\_error& e){

cerr << "\n Invalid entry: " << e.what() << endl;

keep\_window\_open("q");

return 1;}

if(sorter==1){

cout << "\nHere are your possible flight options for the journey "<<airportInfo[airportDepartNo].name<<" to "<<airportInfo[airportArriveNo].name<<", sorted by Journey Time: \n";

if(dirFlights.size() != 0){

for(int y=0; y<(dirFlights.size()-1);y++){

for(int x=0; x<(dirFlights.size()-1);x++){

if(dirFlights[x].time > dirFlights[x+1].time){

dirFlight temp = dirFlights[x];

dirFlights[x] = dirFlights[x+1];

dirFlights[x+1] = temp;}}}

printDirFlights(dirFlights);}

if(conFlights.size() != 0){

for(int y=0; y<(conFlights.size()-1);y++){

for(int x=0; x<(conFlights.size()-1);x++){

int time1 = conFlights[x].time1 + conFlights[x].time2 + conFlights[x].conTime;

int time2 = conFlights[x+1].time1 + conFlights[x+1].time2 + conFlights[x+1].conTime;

if(time1 > time2){

conFlight temp = conFlights[x];

conFlights[x] = conFlights[x+1];

conFlights[x+1] = temp;}}}

printConFlights(conFlights);}

if(twoConFlights.size() != 0){

for(int y=0; y<(twoConFlights.size()-1);y++){

for(int x=0; x<(twoConFlights.size()-1);x++){

int time1 = twoConFlights[x].time1 + twoConFlights[x].time2 + twoConFlights[x].time3 + twoConFlights[x].conTime1 + twoConFlights[x].conTime2;

int time2 = twoConFlights[x+1].time1 + twoConFlights[x+1].time2 + twoConFlights[x+1].time3 + twoConFlights[x+1].conTime1 + twoConFlights[x+1].conTime2;

if(time1 > time2){

twoConFlight temp = twoConFlights[x];

twoConFlights[x] = twoConFlights[x+1];

twoConFlights[x+1] = temp;}}}

printTwoConFlights(twoConFlights);}}

if(sorter==2){

cout << "\nHere are your possible flight options for the journey "<<airportInfo[airportDepartNo].name<<" to "<<airportInfo[airportArriveNo].name<<", sorted by Cost: \n";

if(dirFlights.size() != 0){

for(int y=0; y<(dirFlights.size()-1);y++){

for(int x=0; x<(dirFlights.size()-1);x++){

int cost1 = (dirFlights[x].cost\*(dirFlights[x].tax1)/100)+(dirFlights[x].cost\*(dirFlights[x].tax2)/100) + dirFlights[x].cost;

int cost2 = (dirFlights[x+1].cost\*(dirFlights[x+1].tax1)/100)+(dirFlights[x+1].cost\*(dirFlights[x+1].tax2)/100) + dirFlights[x+1].cost;

if(cost1 > cost2){

dirFlight temp = dirFlights[x];

dirFlights[x] = dirFlights[x+1];

dirFlights[x+1] = temp;}}}

printDirFlights(dirFlights);}

if(conFlights.size() != 0){

for(int y=0; y<(conFlights.size()-1);y++){

for(int x=0; x<(conFlights.size()-1);x++){

int cost1 = (conFlights[x].cost1\*(conFlights[x].tax1)/100)+(conFlights[x].cost1\*(conFlights[x].tax2)/100)+(conFlights[x].cost2\*(conFlights[x].tax2)/100)+(conFlights[x].cost2\*(conFlights[x].tax3)/100) + conFlights[x].cost1 + conFlights[x].cost2;

int cost2 = (conFlights[x+1].cost1\*(conFlights[x+1].tax1)/100)+(conFlights[x+1].cost1\*(conFlights[x+1].tax2)/100)+(conFlights[x+1].cost2\*(conFlights[x+1].tax2)/100)+(conFlights[x+1].cost2\*(conFlights[x+1].tax3)/100) + conFlights[x+1].cost1 + conFlights[x+1].cost2;

if(cost1 > cost2){

conFlight temp = conFlights[x];

conFlights[x] = conFlights[x+1];

conFlights[x+1] = temp;}}}

printConFlights(conFlights);}

if(twoConFlights.size() != 0){

for(int y=0; y<(twoConFlights.size()-1);y++){

for(int x=0; x<(twoConFlights.size()-1);x++){

int cost1 = (twoConFlights[x].cost1\*(twoConFlights[x].tax1)/100)+(twoConFlights[x].cost1\*(twoConFlights[x].tax2)/100)+(twoConFlights[x].cost2\*(twoConFlights[x].tax2)/100)+(twoConFlights[x].cost2\*(twoConFlights[x].tax3)/100)+(twoConFlights[x].cost3\*(twoConFlights[x].tax3)/100)+(twoConFlights[x].cost3\*(twoConFlights[x].tax4)/100)+twoConFlights[x].cost1+twoConFlights[x].cost2+twoConFlights[x].cost3;

int cost2 = (twoConFlights[x+1].cost1\*(twoConFlights[x+1].tax1)/100)+(twoConFlights[x+1].cost1\*(twoConFlights[x+1].tax2)/100)+(twoConFlights[x+1].cost2\*(twoConFlights[x+1].tax2)/100)+(twoConFlights[x+1].cost2\*(twoConFlights[x+1].tax3)/100)+(twoConFlights[x+1].cost3\*(twoConFlights[x+1].tax3)/100)+(twoConFlights[x+1].cost3\*(twoConFlights[x+1].tax4)/100)+twoConFlights[x+1].cost1+twoConFlights[x+1].cost2+twoConFlights[x+1].cost3;

if(cost1 > cost2){

twoConFlight temp = twoConFlights[x];

twoConFlights[x] = twoConFlights[x+1];

twoConFlights[x+1] = temp;}}}

printTwoConFlights(twoConFlights);}}

if(sorter==3){

if(dirFlights.size() != 0){

cout << "\nWhich airline would you to limit your results to: \n 1. Easyjet \n 2. RyanAir \n 3. JetTo \n";

int airNo;

string airChoice;

cin >> airNo;

if(airNo==1){airChoice = "Easyjet";}

if(airNo==2){airChoice = "RyanAir";}

if(airNo==3){airChoice = "JetTo";}

int amount = (dirFlights.size())-1;

for(int x=amount; x>-1; x--){

if(dirFlights[x].airline != airChoice){dirFlights.erase(dirFlights.begin()+(x));}}

printDirFlights(dirFlights);

if(dirFlights.size()==0){cout << "\n There are no flights according to the option you chose \n"; keep\_window\_open("q");return 1;}}}

if(sorter==4){}}

else

{cout << "\nYou have no sorting options as only one flight is availible according to you specification \n";}

int choice;

if(dirFlights.size() != 0){noOptions=dirFlights.size();}

if(conFlights.size() != 0){noOptions=conFlights.size();}

if(twoConFlights.size() != 0){noOptions=twoConFlights.size();}

try{

cout <<"\nPlease choose the flight you want to book by typing in the number next to it and pressing enter: \n";

cin >> choice;

if(choice>noOptions){error("You did not choose a valid number \n");}

}

catch (runtime\_error& e){

cerr << "\nInvalid entry: " << e.what() << endl;

keep\_window\_open("q");

return 1;

}

int flightNo = (choice-1);

//outputting the result on the screen and into a txt file.

//depending on the type of journey chosen, the information and layout of the output changes

if(dirFlights.size() != 0){bookDirFlight(dirFlights, monthStr, day, year, flightNo);}

if(conFlights.size() != 0){bookConFlight(conFlights, monthStr, day, year, flightNo);}

if(twoConFlights.size() != 0){bookTwoConFlight(twoConFlights, monthStr, day, year, flightNo);};

cout << "\n \nCongratulations your flight has now been booked! \n \nYour receipt has been printed under the filename receipt.txt \n \n";

keep\_window\_open("q");

}