**FINAL TERM PROJECT REPORT**

**Made By Rohaan Nadeem**

**PROJECT NAME**

**Title:-** **Airport Management System**

**Objectives:**

=> To add Flight details.

=> To remove Flight details.

=> To add customer details.

=> To remove Flight details.

=> Buy Ticket.

=> Cancel Ticket.

**Note:**

To run it create 2 txt files of customesr.txt, flights.txt

**Code:**

#include <iostream>

#include <windows.h>

#include <fstream>

using namespace std;

int toInt(string name)

{

int num = 0;

int i=0;

while(name[i] != '\0')

{

switch(name[i])

{

case '1':

{

num = num\*10+1;

break;

}

case '2':

{

num = num\*10+2;

break;

}

case '3':

{

num = num\*10+3;

break;

}

case '4':

{

num = num\*10+4;

break;

}

case '5':

{

num = num\*10+5;

break;

}

case '6':

{

num = num\*10+6;

break;

}

case '7':

{

num = num\*10+7;

break;

}

case '8':

{

num = num\*10+8;

break;

}

case '9':

{

num = num\*10+9;

break;

}

case '0':

{

num = num\*10+0;

break;

}

}

i++;

}

return num;

}

//-----------------------------------------------Start Customer Class--------------------------------------------

class Customer

{

private:

int customerId;

string name;

string phone;

string date\_of\_birth;

int noOfFlights;

int milesTraveled;

Customer \*Left;

Customer \*Right;

public:

Customer()

{

customerId = 0;

name = "";

phone = "";

date\_of\_birth = "";

noOfFlights = 0;

milesTraveled = 0;

Left = NULL;

Right = NULL;

}

void setCustomerId(int customerId)

{

this->customerId = customerId;

}

void setLeft(Customer \*Left)

{

this->Left = Left;

}

void setRight(Customer \*Right)

{

this->Right = Right;

}

void setName(string name)

{

this->name = name;

}

void setPhone(string phone)

{

this->phone = phone;

}

void setDateOfBirth(string date\_of\_birth)

{

this->date\_of\_birth = date\_of\_birth;

}

void setNoOfFlights(int noOfFlights)

{

this->noOfFlights = noOfFlights;

}

void setMilesTraveled(int milesTraveled)

{

this->milesTraveled = milesTraveled;

}

int getCustomerId()

{

return customerId;

}

string getName()

{

return name;

}

string getPhone()

{

return phone;

}

string getDateOfBirth()

{

return date\_of\_birth;

}

int getNoOfFlights()

{

return noOfFlights;

}

int getMilesTraveled()

{

return milesTraveled;

}

Customer \*getLeft()

{

return Left;

}

Customer \*getRight()

{

return Right;

}

void printCustomer()

{

cout<<"CUSTOMER ID IS : "<<customerId<<endl;

cout<<"CUSTOMER NAME IS : "<<name<<endl;

cout<<"CUSTOMER PHONE NUMBER IS : "<<phone<<endl;

cout<<"CUSTOMER FLIGHT NUMBER IS : "<<noOfFlights<<endl;

cout<<"CUSTOMER DATE OF BIRTH IS : "<<date\_of\_birth<<endl;

cout<<"CUSTOMER TRAVELED MILES ARE : "<<milesTraveled<<endl;

}

};

class BinarySearchTree

{

private:

int id;

public:

Customer \*root;

BinarySearchTree()

{

root = NULL;

id = 1;

}

void insertData()

{

Customer \*customer = new Customer;

cout<<" PLEASE ENTER THE INFORMATION "<<endl;

string name;

string phone;

string date\_of\_birth;

fflush(stdin);

cout<<"ENTER CUSTOMER NAME : ";

getline(cin,name);

fflush(stdin);

cout<<"ENTER CUSTOMER PHONE NUMBER : ";

getline(cin,phone);

fflush(stdin);

cout<<"ENTER CUSTOMER DATE OF BIRTH : ";

getline(cin,date\_of\_birth);

customer->setDateOfBirth(date\_of\_birth);

customer->setName(name);

customer->setPhone(phone);

customer->setCustomerId(id++);

if(root == NULL)

{

root = customer;

}

else

{

Customer \*temp = root;

Customer \*prev = root;

while(temp != NULL)

{

prev = temp;

if(customer->getCustomerId() < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

if(customer->getCustomerId() < prev->getCustomerId())

{

prev->setLeft(customer);

}

else

{

prev->setRight(customer);

}

}

}

void searchCustomer(int CustomerId)

{

if(root == NULL)

{

cout<<"NO CUSTOMER ADDED YET"<<endl;

}

else

{

bool flag = true;

Customer \*temp = root;

while(temp != NULL)

{

if(CustomerId == temp->getCustomerId())

{

flag = false;

temp->printCustomer();

return;

}

else if(CustomerId < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

if(flag)

{

cout<<"DATA NOT FOUND"<<endl;

}

else

{}

}

}

bool getCustomerExistance(int CustomerId)

{

if(root == NULL)

{

cout<<"NO CUSTOMER ADDED YET"<<endl;

}

else

{

bool flag = true;

Customer \*temp = root;

while(temp != NULL)

{

if(CustomerId == temp->getCustomerId())

{

flag = false;

return true;

}

else if(CustomerId < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

if(flag)

{

cout<<"DATA NOT FOUND"<<endl;

return false;

}

else

{}

}

}

void updateCustomerTravelInfo(int dis,int CustomerId)

{

if(root == NULL)

{

cout<<"NO CUSTOMER ADDED YET"<<endl;

}

else

{

Customer \*temp = root;

while(temp != NULL)

{

if(CustomerId == temp->getCustomerId())

{

if(dis<0)

{

temp->setNoOfFlights(temp->getNoOfFlights()-1);

temp->setMilesTraveled(temp->getMilesTraveled()+dis);

}

else

{

temp->setNoOfFlights(temp->getNoOfFlights()+1);

temp->setMilesTraveled(temp->getMilesTraveled()+dis);

}

return;

}

else if(CustomerId < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

}

}

string getCustomerName(int CustomerId)

{

if(root == NULL)

{

cout<<"NO CUSTOMER ADDED YET"<<endl;

}

else

{

bool flag = true;

Customer \*temp = root;

while(temp != NULL)

{

if(CustomerId == temp->getCustomerId())

{

flag = false;

return temp->getName();

}

else if(CustomerId < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

if(flag)

{

cout<<"DATA NOT FOUND"<<endl;

}

else

{}

}

}

void updateCustomer(int CustomerId)

{

if(root == NULL)

{

cout<<"NO CUSTOMER ADDED YET"<<endl;

}

else

{

bool flag = true;

Customer \*temp = root;

while(temp != NULL)

{

if(CustomerId == temp->getCustomerId())

{

flag = false;

cout<<" PLEASE UPDATE THE INFORMATION "<<endl;

string name;

string phone;

string date\_of\_birth;

fflush(stdin);

cout<<"ENTER CUSTOMER NAME : ";

getline(cin,name);

fflush(stdin);

cout<<"ENTER CUSTOMER PHONE NUMBER : ";

getline(cin,phone);

fflush(stdin);

cout<<"ENTER CUSTOMER DATE OF BIRTH : ";

getline(cin,date\_of\_birth);

temp->setDateOfBirth(date\_of\_birth);

temp->setName(name);

temp->setPhone(phone);

return;

}

else if(CustomerId < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

if(flag)

{

cout<<"DATA NOT FOUND"<<endl;

}

else

{

cout<<"DATA FOUNDED SUCCESSFULLY"<<endl;

}

}

}

void loadData()

{

ifstream load;

load.open("customers.txt");

while(!load.eof())

{

Customer \*customer = new Customer;

string line;

getline(load,line);

if(line=="")

continue;

int i=0;

string stemp="";

while(line[i]!='@')

{

stemp +=line[i];

i++;

}

customer->setCustomerId(toInt(stemp));

i++;

stemp="";

while(line[i]!='@')

{

stemp +=line[i];

i++;

}

customer->setName(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp +=line[i];

i++;

}

customer->setPhone(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp +=line[i];

i++;

}

customer->setDateOfBirth(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp +=line[i];

i++;

}

customer->setNoOfFlights(toInt(stemp));

i++;

stemp="";

while(line[i]!='\0')

{

stemp +=line[i];

i++;

}

customer->setMilesTraveled(toInt(stemp));

if(root == NULL)

{

root = customer;

}

else

{

Customer \*temp = root;

Customer \*prev = root;

while(temp != NULL)

{

prev = temp;

if(customer->getCustomerId() < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

if(customer->getCustomerId() < prev->getCustomerId())

{

prev->setLeft(customer);

}

else

{

prev->setRight(customer);

}

}

id++;

}

}

void saveData(Customer \*temp)

{

if(temp==NULL)

{

return;

}

ofstream Input;

Input.open("customers.txt",ios::app);

Input<<temp->getCustomerId()<<"@";

Input<<temp->getName()<<"@";

Input<<temp->getPhone()<<"@";

Input<<temp->getDateOfBirth()<<"@";

Input<<temp->getNoOfFlights()<<"@";

Input<<temp->getMilesTraveled()<<endl;

saveData(temp->getLeft());

saveData(temp->getRight());

}

};

//-----------------------------------------------End Customer Class--------------------------------------------

//-----------------------------------------------Start Airport Class--------------------------------------------

class Airport

{

private:

string airportName;

string country;

string usualWeatherCondition;

int timeFromBeirut;

int milesFromBeirut;

public:

Airport()

{

airportName = "";

country = "";

usualWeatherCondition = "";

timeFromBeirut = 0;

milesFromBeirut = 0;

}

void setAirportName(string airportName)

{

this->airportName = airportName;

}

void setCountry(string country)

{

this->country = country;

}

void setUsualWeatherCondition(string usualWeatherCondition)

{

this->usualWeatherCondition = usualWeatherCondition;

}

void setTimeFromBeirut(int timeFromBeirut)

{

this->timeFromBeirut = timeFromBeirut;

}

void setMilesFromBeirut(int milesFromBeirut)

{

this->milesFromBeirut = milesFromBeirut;

}

string getAirportName()

{

return airportName;

}

string getCountry()

{

return country;

}

string getUsualWeatherCondition()

{

return usualWeatherCondition;

}

int getTimeFromBeirut()

{

return timeFromBeirut;

}

int getMilesFromBeirut()

{

return milesFromBeirut;

}

void display()

{

cout<<"AIRPORT NAME IS : "<<airportName<<endl;

cout<<"COUNTRY IS : "<<country<<endl;

cout<<"WEARHER CONDITION IS : "<<usualWeatherCondition<<endl;

}

};

//-----------------------------------------------End Airport Class--------------------------------------------

class Ticket

{

private:

int customerId;

string name;

int flightNumber;

int ticketNumber;

string ticketType;

bool discount;

string discountType;

string dateTime;

int seatNumber;

Ticket \*left;

Ticket \*right;

Ticket \*Next;

public:

Ticket()

{

customerId = 0;

name = "";

flightNumber = 0;

ticketNumber = 0;

ticketType = "";

discount = false;

discountType = "";

dateTime = "";

seatNumber = 0;

Next = NULL;

left = NULL;

right = NULL;

}

void setCustomerId(int customerId)

{

this->customerId = customerId;

}

void setName(string name)

{

this->name = name;

}

void setFlightNumber(int flightNumber)

{

this->flightNumber = flightNumber;

}

void setTicketNumber(int ticketNumber)

{

this->ticketNumber = ticketNumber;

}

void setTicketType(string ticketType)

{

this->ticketType = ticketType;

}

void setDiscount(bool discount)

{

this->discount = discount;

}

void setDiscountType(string discountType)

{

this->discountType = discountType;

}

void setDateTime(string dateTime)

{

this->dateTime = dateTime;

}

void setSeatNumber(int seatNumber)

{

this->seatNumber = seatNumber;

}

void setLeft(Ticket \*left)

{

this->left = left;

}

void setRight(Ticket \*right)

{

this->right = right;

}

void setNext(Ticket \*Next)

{

this->Next = Next;

}

int getCustomerId()

{

return customerId;

}

string getName()

{

return name;

}

int getFlightNumber()

{

return flightNumber;

}

int getTicketNumber()

{

return ticketNumber;

}

string getTicketType()

{

return ticketType;

}

bool getDiscount()

{

return discount;

}

string getDiscountType()

{

return discountType;

}

string getDateTime()

{

return dateTime;

}

int getSeatNumber()

{

return seatNumber;

}

Ticket \*getLeft()

{

return left;

}

Ticket \*getRight()

{

return right;

}

Ticket \*getNext()

{

return Next;

}

void print()

{

cout<<"CUSTOMER ID IS : "<<customerId<<endl;

cout<<"NAME IS : "<<name<<endl;

cout<<"FLIGHT NUMBER IS : "<<flightNumber<<endl;

cout<<"TICKET NUMBER IS : "<<ticketNumber<<endl;

cout<<"TICKET TYPE IS : "<<ticketType<<endl;

cout<<"DISCOUNT TYPE IS : "<<discountType<<endl;

cout<<"DATE OR TIME IS : "<<dateTime<<endl;

cout<<"SEAT NUMBER IS : "<<seatNumber<<endl;

}

};

class PassengerQueue

{

private:

Ticket \*head;

public:

PassengerQueue()

{

head = NULL;

}

void enqueue(Ticket passenger)

{

Ticket \*newTicket = new Ticket;

newTicket->setCustomerId(passenger.getCustomerId());

newTicket->setDiscountType(passenger.getDiscountType());

newTicket->setName(passenger.getName());

newTicket->setFlightNumber(passenger.getFlightNumber());

newTicket->setTicketNumber(passenger.getTicketNumber());

newTicket->setTicketType(passenger.getTicketType());

newTicket->setDiscount(passenger.getDiscount());

newTicket->setDateTime(passenger.getDateTime());

newTicket->setSeatNumber(passenger.getSeatNumber());

if(head == NULL)

{

head = newTicket;

}

else

{

Ticket \*temp = head;

while(temp->getNext()!=NULL)

{

temp = temp->getNext();

}

temp->setNext(newTicket);

}

}

Ticket dequeue()

{

if(head == NULL)

{

cout<<"QUEUE IS EMPTY PLEASE!!!!"<<endl;

}

else

{

Ticket \*temp = head;

head = head->getNext();

temp->setNext(NULL);

Ticket result = \*temp;

delete temp;

return result;

}

}

bool isEmpty()

{

if(head == NULL)

{

return true;

}

return false;

}

};

class PassengerBST

{

private:

public:

Ticket \*root;

PassengerBST()

{

root = NULL;

}

void insert(Ticket passenger)

{

Ticket \*newTicket = new Ticket;

newTicket->setCustomerId(passenger.getCustomerId());

newTicket->setDiscountType(passenger.getDiscountType());

newTicket->setName(passenger.getName());

newTicket->setFlightNumber(passenger.getFlightNumber());

newTicket->setTicketNumber(passenger.getTicketNumber());

newTicket->setTicketType(passenger.getTicketType());

newTicket->setDiscount(passenger.getDiscount());

newTicket->setDateTime(passenger.getDateTime());

newTicket->setSeatNumber(passenger.getSeatNumber());

newTicket->setLeft(NULL);

newTicket->setRight(NULL);

if(root == NULL)

{

root = newTicket;

}

else

{

Ticket \*temp = root;

Ticket \*prev = root;

while(temp!=NULL)

{

prev = temp;

if(passenger.getCustomerId() < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

if(passenger.getCustomerId() < prev->getCustomerId())

{

prev->setLeft(newTicket);

}

else

{

prev->setRight(newTicket);

}

}

}

void display(Ticket \*temp)

{

if(temp == NULL)

{

return;

}

temp->print();

display(temp->getLeft());

display(temp->getRight());

}

bool PassengerExistanceInFlight(int id)

{

Ticket \*temp = root;

while(temp!=NULL)

{

if(id == temp->getCustomerId())

{

return false;

}

else if(id < temp->getCustomerId())

{

temp = temp->getLeft();

}

else

{

temp = temp->getRight();

}

}

return true;

}

void deletePassanger(int cid)

{

Ticket \*temp = root;

Ticket \*p = root;

int compare;

if (root == NULL)

{

return;

}

while (temp->getCustomerId() != cid)

{

if(temp->getCustomerId() == cid)

{

compare = 0;

}

else

{

compare = 1;

}

p = temp;

if (compare > 0 && temp->getLeft() != NULL)

{

temp = temp->getLeft();

}

else if (compare <= 0 && temp->getRight() != NULL)

{

temp = temp->getRight();

}

else

{

break;

}

}

if (temp->getCustomerId() == cid)

{

if (root == temp && root->getLeft() == NULL && root->getRight() == NULL)

{

root = NULL;

}

else if (root == temp && temp->getLeft() != NULL && temp->getRight() == NULL)

{

root = temp->getLeft();

}

else if (root == temp && temp->getLeft() == NULL && temp->getRight() != NULL)

{

root = temp->getRight();

}

else if (temp->getLeft() == NULL && temp->getRight() == NULL)

{

if (p->getLeft() == temp)

{

p->setLeft(NULL);

}

else if (p->getRight() == temp)

{

p->setRight(NULL);

}

}

else if ((temp->getLeft() != NULL && temp->getRight() == NULL) || (temp->getLeft() == NULL && temp->getRight() != NULL))

{

if (temp->getLeft() != NULL)

{

p->setLeft(temp->getLeft());

}

else if (temp->getRight() != NULL)

{

p->setRight(temp->getRight());

}

}

else if (temp->getRight() != NULL && temp->getLeft() != NULL)

{

Ticket \*q = temp->getRight();

Ticket \*parent\_q = q;

while (q->getLeft() != NULL)

{

parent\_q = q;

q = q->getLeft();

}

if ((parent\_q->getLeft() == NULL && parent\_q->getRight() == NULL) && (q->getLeft() == NULL && q->getRight() == NULL))

{

temp->setName(q->getName());

temp->setDateTime(q->getDateTime());

temp->setFlightNumber(q->getFlightNumber());

temp->setSeatNumber(q->getSeatNumber());

temp->setTicketNumber(q->getTicketNumber());

temp->setTicketType(q->getTicketType());

temp->setRight(NULL);

}

else

{

parent\_q->setLeft(q->getRight());

temp->setName(q->getName());

temp->setCustomerId(q->getCustomerId());

temp->setDateTime(q->getDateTime());

temp->setFlightNumber(q->getFlightNumber());

temp->setSeatNumber(q->getSeatNumber());

temp->setTicketNumber(q->getTicketNumber());

temp->setTicketType(q->getTicketType());

}

}

}

else

{

cout << "Name Does Not Exists !!!" << endl;

}

}

};

//-----------------------------------------------Start Flight Class--------------------------------------------

class Flight

{

private:

int flightNumber;

int planeId;

int day;

int month;

int year;

string departureTime;

Airport destination;

PassengerBST passengers;

double distance;

Flight \*Next;

Flight \*Prev;

int no\_of\_tickets;

PassengerQueue waitingPassenger;

public:

Flight()

{

no\_of\_tickets=0;

flightNumber = 0;

planeId = 0;

day = 0;

month = 0;

year = 0;

distance = 0;

Next = NULL;

Prev = NULL;

}

void insertWaitingPassanger(int cid,string cname,int fid,string fclass)

{

Ticket obj;

obj.setCustomerId(cid);

obj.setName(cname);

obj.setFlightNumber(fid);

obj.setTicketType(fclass);

obj.setTicketNumber(no\_of\_tickets);

obj.setDateTime(getDepartureTime());

obj.setSeatNumber(no\_of\_tickets+1);

obj.setTicketNumber(no\_of\_tickets+1);

waitingPassenger.enqueue(obj);

}

bool isEmpty()

{

if(no\_of\_tickets==3)

return false;

return true;

}

void setFlightNumber(int flightNumber)

{

this->flightNumber = flightNumber;

}

void setPlaneId(int planeId)

{

this->planeId = planeId;

}

void setDepartureTime(string departureTime)

{

this->departureTime = departureTime;

}

void setDay(int day)

{

this->day = day;

}

void setMonth(int month)

{

this->month = month;

}

void setYear(int year)

{

this->year = year;

}

void setDistance(double distance)

{

this->distance = distance;

}

void setDestination(Airport destination)

{

this->destination = destination;

}

void insertPassengers(int cid,string cname,int fid,string fclass)

{

Ticket obj;

obj.setCustomerId(cid);

obj.setName(cname);

obj.setFlightNumber(fid);

obj.setTicketType(fclass);

obj.setTicketNumber(no\_of\_tickets);

obj.setDateTime(getDepartureTime());

obj.setSeatNumber(no\_of\_tickets+1);

obj.setTicketNumber(no\_of\_tickets+1);

passengers.insert(obj);

no\_of\_tickets++;

}

void deletePassengers(int cid)

{

passengers.deletePassanger(cid);

if(waitingPassenger.isEmpty())

{

no\_of\_tickets--;;

}

else

{

passengers.insert(waitingPassenger.dequeue());

}

}

void displayPassengers()

{

passengers.display(passengers.root);

}

void setNext(Flight \*Next)

{

this->Next = Next;

}

void setPrev(Flight \*Prev)

{

this->Prev = Prev;

}

int getFlightNumber()

{

return flightNumber;

}

int getPlaneId()

{

return planeId;

}

string getDepartureTime()

{

return departureTime;

}

int getDay()

{

return day;

}

int getMonth()

{

return month;

}

int getYear()

{

return year;

}

double getDistance()

{

return distance;

}

Airport getAirportDestination()

{

return destination;

}

Flight \*getNext()

{

return Next;

}

Flight \*getPrev()

{

return Prev;

}

void display()

{

cout<<"FLIGHT NUMBER IS : "<<flightNumber<<endl;

cout<<"PLANE ID IS : "<<planeId<<endl;

cout<<"FLIGHT DAY IS : "<<day<<endl;

cout<<"FLIGHT MONTH IS : "<<month<<endl;

cout<<"FLIGHT YEAR IS : "<<year<<endl;

cout<<"DEPARTURE TIME IS : "<<departureTime<<endl;

cout<<"AIRPORT DESTINATION :"<<endl;

destination.display();

}

};

class FlightDoublyList

{

private:

Flight \*head;

public:

FlightDoublyList()

{

head = NULL;

}

bool isEmpty(int cid,string cname,int id,string fclass)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY"<<endl;

}

else

{

Flight \*temp = head;

while (temp!=NULL)

{

if(temp->getPlaneId() == id)

{

bool answer = temp->isEmpty();

if(!answer)

{

cout<<"NOW FLIGHT IS FULL"<<endl;

cout<<"YOU WANT TO WAIT FOR IT (y/n): ";

char ch;

cin>>ch;

if(ch=='y' || ch=='Y')

{

temp->insertWaitingPassanger(cid,cname,id,fclass);

return answer;

}

else

{

return answer;

}

}

else

{

return answer;

}

}

temp = temp->getNext();

}

}

}

bool flightExistance(int id)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY"<<endl;

}

else

{

Flight \*temp = head;

while (temp!=NULL)

{

if(temp->getPlaneId() == id)

{

return true;

}

temp = temp->getNext();

}

return false;

}

}

void load()

{

ifstream load;

load.open("flights.txt");

while(!load.eof())

{

Flight \*newNode = new Flight;

Airport dest;

string line;

getline(load,line);

if(line=="")

continue;

int i=0;

string stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

newNode->setFlightNumber(toInt(stemp));

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

newNode->setPlaneId(toInt(stemp));

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

newNode->setDay(toInt(stemp));

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

newNode->setMonth(toInt(stemp));

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

newNode->setYear(toInt(stemp));

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

newNode->setDepartureTime(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

dest.setAirportName(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

dest.setCountry(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

i++;

}

dest.setUsualWeatherCondition(stemp);

newNode->setDestination(dest);

i++;

stemp="";

while(line[i]!='\0')

{

stemp += line[i];

i++;

}

newNode->setDistance(toInt(stemp));

newNode->setNext(NULL);

newNode->setPrev(NULL);

if(head == NULL)

{

head = newNode;

}

else

{

newNode->setNext(head);

head->setPrev(newNode);

head = newNode;

}

}

}

int getFlightDistance(int id)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY"<<endl;

}

else

{

Flight \*temp = head;

while (temp!=NULL)

{

if(temp->getPlaneId() == id)

{

return temp->getDistance();

}

temp = temp->getNext();

}

}

}

void insertFlight()

{

Flight \*newNode = new Flight;

Airport dest;

int fn,day,month,year,dist,pid;

string dt,an,country,uwc;

fflush(stdin);

cout<<"ENTER FLIGHT NUMBER : ";

cin>>fn;

fflush(stdin);

cout<<"ENTER PLANE I'D : ";

cin>>pid;

fflush(stdin);

cout<<"ENTER DATE OF FLIGHT : ";

cin>>day;

fflush(stdin);

cout<<"ENTER MONTH OF FLIGHT : ";

cin>>month;

fflush(stdin);

cout<<"ENTER YEAR OF FLIGHT : ";

cin>>year;

fflush(stdin);

cout<<"ENTER DISTANCE OF FLIGHT : ";

cin>>dist;

fflush(stdin);

cout<<"ENTER DEPARTURE ( TIME ) OF FLIGHT : ";

getline(cin,dt);

fflush(stdin);

cout<<"ENTER THE DESTINATION AIRPORT NAME : ";

getline(cin,an);

fflush(stdin);

cout<<"ENTER THE COUNTRY WHERE FLIGHT IS GOING : ";

getline(cin,country);

fflush(stdin);

cout<<"ENTER THE WEATHER CONDITION : ";

getline(cin,uwc);

newNode->setFlightNumber(fn);

newNode->setPlaneId(pid);

newNode->setDay(day);

newNode->setMonth(month);

newNode->setYear(year);

newNode->setDepartureTime(dt);

dest.setAirportName(an);

dest.setCountry(country);

dest.setUsualWeatherCondition(uwc);

newNode->setDestination(dest);

newNode->setDistance(dist);

newNode->setNext(NULL);

newNode->setPrev(NULL);

if(head == NULL)

{

head = newNode;

}

else

{

newNode->setNext(head);

head->setPrev(newNode);

head = newNode;

}

}

void saveData()

{

if(head == NULL)

{

cout<<"LIST IS EMPTY";

}

else

{

Flight \*temp = head;

ofstream input;

input.open("Flights.txt");

while (temp!=NULL)

{

input<<temp->getFlightNumber()<<"@"<<temp->getPlaneId()<<"@"<<temp->getDay();

input<<"@"<<temp->getMonth()<<"@"<<temp->getYear()<<"@"<<temp->getDepartureTime();

input<<"@"<<temp->getAirportDestination().getAirportName()<<"@"<<temp->getAirportDestination().getCountry();

input<<"@"<<temp->getAirportDestination().getUsualWeatherCondition()<<"@"<<temp->getDistance()<<endl;

temp = temp->getNext();

}

input.close();

}

}

void updateFlight(int pid)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY";

}

else

{

Flight \*temp = head;

while (temp!=NULL)

{

if(temp->getPlaneId() == pid)

{

Airport dest;

int fn,day,month,year,dist,id;

string dt,an,country,uwc;

fflush(stdin);

cout<<"ENTER FLIGHT NUMBER : ";

cin>>fn;

fflush(stdin);

cout<<"ENTER PLANE I'D : ";

cin>>id;

fflush(stdin);

cout<<"ENTER DAY OF FLIGHT : ";

cin>>day;

fflush(stdin);

cout<<"ENTER MONTH OF FLIGHT : ";

cin>>month;

fflush(stdin);

cout<<"ENTER YEAR OF FLIGHT : ";

cin>>year;

fflush(stdin);

cout<<"ENTER DISTANCE OF FLIGHT : ";

cin>>dist;

fflush(stdin);

cout<<"ENTER DEPARTURE OF FLIGHT : ";

getline(cin,dt);

fflush(stdin);

cout<<"ENTER AIRPORT NAME : ";

getline(cin,an);

fflush(stdin);

cout<<"ENTER THE COUNTRY WHERE FLIGHT IS GOING : ";

getline(cin,country);

fflush(stdin);

cout<<"ENTER THE WEATHER CONDITION : ";

getline(cin,uwc);

temp->setFlightNumber(fn);

temp->setPlaneId(id);

temp->setDay(day);

temp->setMonth(month);

temp->setYear(year);

temp->setDepartureTime(dt);

dest.setAirportName(an);

dest.setCountry(country);

dest.setUsualWeatherCondition(uwc);

temp->setDestination(dest);

temp->setDistance(dist);

}

}

}

}

void deleteFlight(int pid)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY";

}

else

{

if(head->getPlaneId() == pid)

{

if(head->getPlaneId() == pid && head->getNext() == NULL)

{

head = NULL;

}

else

{

Flight \*ptr = head;

head=head->getNext();

head->setPrev(NULL);

ptr->setNext(NULL);

ptr->setPrev(NULL);

delete ptr;

}

}

else

{

Flight \*temp = head;

while(temp!=NULL)

{

if(temp->getNext()->getPlaneId() == pid && temp->getNext()->getNext() == NULL)

{

Flight \*ptr = temp->getNext();

temp->setNext(NULL);

ptr->setNext(NULL);

ptr->setPrev(NULL);

delete ptr;

}

else if(temp->getNext()->getPlaneId() == pid)

{

Flight \*ptr = temp->getNext();

temp->setNext(temp->getNext()->getNext());

temp->getNext()->setPrev(temp);

ptr->setNext(NULL);

ptr->setPrev(NULL);

delete ptr;

break;

}

temp = temp->getNext();

}

}

}

}

void displayFlight(int pid)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY";

}

else

{

Flight \*temp = head;

while(temp!=NULL)

{

if(temp->getPlaneId() == pid)

{

temp->display();

cout<<"PASSENGERS INFORMATION OF THAT FLIGHT IS "<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

temp->displayPassengers();

break;

}

temp = temp->getNext();

}

}

}

void searchFlight(string airportName)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY"<<endl;

}

else

{

Flight \*temp = head;

while (temp!=NULL)

{

if(temp->getAirportDestination().getAirportName() == airportName)

{

temp->display();

}

temp = temp->getNext();

}

}

}

void insertPassenger(int cid,string cname,int fid,string fclass)

{

if(head == NULL)

{

cout<<"LIST IS EMPTY"<<endl;

}

else

{

Flight \*temp = head;

while (temp!=NULL)

{

if(temp->getPlaneId() == fid)

{

temp->insertPassengers(cid,cname,fid,fclass);

}

temp = temp->getNext();

}

}

}

void deletePassenger(int planeid,int customerid )

{

if(head == NULL)

{

cout<<"LIST IS EMPTY"<<endl;

}

else

{

Flight \*temp = head;

while (temp!=NULL)

{

if(planeid == temp->getPlaneId())

{

temp->deletePassengers(customerid);

}

temp = temp->getNext();

}

}

}

};

//-----------------------------------------------End Flight Class--------------------------------------------

//-----------------------------------------------Start Plane Class--------------------------------------------

class Plane

{

private:

int planeId;

string description;

int numberOfMilesTraveled;

int \*numberOfSeatsInBusinessClass;

int \*numberOfSeatsInEconomicalClass;

int size1;

int size2;

Plane \*Next;

public:

Plane(int planeId,string description,int numberOfMilesTraveled,int size1,int size2)

{

this->planeId = planeId;

this->description = description;

this->numberOfMilesTraveled = numberOfMilesTraveled;

this->size1 = size1;

this->size2 = size2;

numberOfSeatsInBusinessClass = new int[size1];

numberOfSeatsInEconomicalClass = new int[size2];

int counter = 1;

for (int i=0; i<size1; i++)

{

numberOfSeatsInBusinessClass[i] = counter;

counter++;

}

for (int i=0; i<size2; i++)

{

numberOfSeatsInEconomicalClass[i] = counter;

counter++;

}

Next = NULL;

}

void setPlaneId(int planeId)

{

this->planeId = planeId;

}

void setDescription(string description)

{

this->description = description;

}

void setNumberOfMilesTraveled(int numberOfMilesTraveled)

{

this->numberOfMilesTraveled = numberOfMilesTraveled;

}

void setNext(Plane \*Next)

{

this->Next = Next;

}

int getPlaneId()

{

return planeId;

}

string getDescription()

{

return description;

}

int getNumberOfMilesTraveled()

{

return numberOfMilesTraveled;

}

int returnBusiness(int i)

{

return numberOfSeatsInBusinessClass[i];

}

int returnEconomical(int i)

{

return numberOfSeatsInEconomicalClass[i];

}

Plane \*getNext()

{

return Next;

}

void printPlane()

{

cout<<"PLANE ID IS : "<<planeId<<endl;

cout<<"PLANE DESCRIPTION IS : "<<description<<endl;

cout<<"PLANE NUMBER OF MILES TRAVELED BY PLAIN ARE : "<<numberOfMilesTraveled<<endl;

cout<<"PLANE BUSINESS CLASS SEATS ARE : ";

for (int i=0; i<size1; i++)

{

cout<<numberOfSeatsInBusinessClass[i]<<" ";

}

cout<<"PLANE ECONOMIC CLASS SEATS ARE : ";

for (int i=0; i<size2; i++)

{

cout<<numberOfSeatsInEconomicalClass[i]<<" ";

}

}

};

class PlaneList

{

private:

Plane \*head;

public:

PlaneList()

{

head = NULL;

}

void load()

{

ifstream load;

load.open("planes.txt");

while(!load.eof())

{

int PlaneId,numberOfMilesTraveled,size1,size2;

string description,line;

getline(load,line);

int i=0;

string stemp="";

while(line[i]!='@')

{

stemp += line[i];

}

PlaneId = toInt(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

}

description = stemp;

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

}

numberOfMilesTraveled = toInt(stemp);

i++;

stemp="";

while(line[i]!='@')

{

stemp += line[i];

}

size1 = toInt(stemp);

i++;

stemp="";

while(line[i]!='\0')

{

stemp += line[i];

}

size2 = toInt(stemp);

Plane \*newPlane = new Plane(PlaneId,description,numberOfMilesTraveled,size1,size2);

newPlane->setNext(NULL);

if(head == NULL)

{

head = newPlane;

}

else

{

Plane \*temp = head;

while(temp->getNext()!=NULL)

{

temp = temp->getNext();

}

temp->setNext(newPlane);

}

}

}

void searchPlane(int PlaneId)

{

if(head == NULL)

{

cout<<"NO PLANE IS READY YET"<<endl;

}

else

{

Plane \*temp = head;

while(temp->getNext()!=NULL)

{

if(PlaneId == temp->getPlaneId())

{

temp->printPlane();

}

temp = temp->getNext();

}

}

}

};

//-----------------------------------------------End Plane Class--------------------------------------------

//-----------------------------------------------End of all Classes--------------------------------------------

//-----------------------------------------------Start Menu Function--------------------------------------------

void outlineMenu()

{

cout<<"\t\t\t\t\t\t\t\tPRESS 1 TO GO TO MENU"<<endl;

cout<<"\t\t\t\t\t\t\t\tPRESS 2 TO EXIT"<<endl;

cout<<"\t\t\t\t\t\t\t\tENTER YOUR OPTION NOW!!!! : ";

}

void furtherMenu()

{

cout<<"\t\t\t\t\t\t\t\t -:Welcome to Menu:- "<<endl<<endl;

cout<<"\t\t\t\t\t\t\t\tPRESS 1 TO GO TO FLIGHT RESERVATION MENU"<<endl;

cout<<"\t\t\t\t\t\t\t\tPRESS 2 TO GO TO CUSTOMER MENU"<<endl;

cout<<"\t\t\t\t\t\t\t\tPRESS 3 TO GO TO FLIGHT MENU"<<endl;

cout<<"\t\t\t\t\t\t\t\tPRESS 4 TO SAVE DATA TO TEXT FILES"<<endl;

cout<<"\t\t\t\t\t\t\t\tPRESS 5 TO GO BACK"<<endl;

cout<<"\t\t\t\t\t\t\t\tENTER YOUR OPTION NOW!!!! : ";

}

void flightReservationMenu()

{

cout<<"\t\t\t\t\t\t\t\t\tFLIGHT RESERVATION"<<endl<<endl;

cout<<"\t\t\t\t\t\t\t\ta.SEARCH FOR DESTINATION DATA"<<endl;

cout<<"\t\t\t\t\t\t\t\tb.RESERVE A FLIGHT"<<endl;

cout<<"\t\t\t\t\t\t\t\tc.CANCEL RESERVATION: UPDATE CUSTOMER INFO,FLIGHT INFO,AND WAITING LIST"<<endl;

cout<<"\t\t\t\t\t\t\t\td.TO GO BACK"<<endl;

cout<<"\t\t\t\t\t\t\t\tENTER YOUR OPTION NOW!!!! : ";

}

void customersMenu()

{

cout<<"\t\t\t\t\t\t\t\t\tCUSTOMERS"<<endl<<endl;

cout<<"\t\t\t\t\t\t\t\ta.ADD CUSTOMER"<<endl;

cout<<"\t\t\t\t\t\t\t\tb.UPDATE CUSTOMER INFO: NOT MILEAGE OR NUMBER OF FLIGHTS"<<endl;

cout<<"\t\t\t\t\t\t\t\tc.VIEW CUSTOMER PROFILE"<<endl;

cout<<"\t\t\t\t\t\t\t\td.TO GO BACK"<<endl;

cout<<"\t\t\t\t\t\t\t\tENTER YOUR OPTION NOW!!!! : ";

}

void flightsMenu()

{

cout<<"\t\t\t\t\t\t\t\t\tFLIGHTS"<<endl<<endl;

cout<<"\t\t\t\t\t\t\t\ta.ADD FLIGHT"<<endl;

cout<<"\t\t\t\t\t\t\t\tb.UPDATE FLIGHT"<<endl;

cout<<"\t\t\t\t\t\t\t\tc.DELETE FLIGHT"<<endl;

cout<<"\t\t\t\t\t\t\t\td.VIEW FLIGHTS"<<endl;

cout<<"\t\t\t\t\t\t\t\te.TO GO BACK"<<endl;

cout<<"\t\t\t\t\t\t\t\tENTER YOUR OPTION NOW!!!! : ";

}

//-----------------------------------------------End Menu Function--------------------------------------------

void timeWaste(int a)

{

for (int i=0; i<=a\*a\*a ; i++)

{

}

}

void loading()

{

char ch=222;

cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;

cout<<"\t\t\t\t\t\t\t\t LOADING!!!!"<<endl;

cout<<"\t\t\t\t\t\t\t\t";

for (int i=1; i<17 ; i++)

{

cout<<ch;

timeWaste(300);

}

}

//-----------------------------------------------Start Main--------------------------------------------

int main()

{

char ch=177;

BinarySearchTree customer;

customer.loadData();

FlightDoublyList flights;

flights.load();

loading();

while(1)

{

Menulabel:

system("cls");

cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<"\t\t\t\t\t\t\t -:{ CREATED BY ROHAAN NADEEM }:-"<<endl<<endl;

cout<<"\t\t\t\t\t\t\t --------------------------------------------"<<endl;

cout<<"\t\t\t\t\t\t\t | |"<<endl;

cout<<"\t\t\t\t\t\t\t | Welcome To Flight Management System |"<<endl;

cout<<"\t\t\t\t\t\t\t | |"<<endl;

cout<<"\t\t\t\t\t\t\t --------------------------------------------"<<endl<<endl;

char opMenu;

outlineMenu();

cin>>opMenu;

switch(opMenu)

{

case '1':

{

while(1)

{

system("cls");

furtherMenulabel:

system("cls");

cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;

char opFurtherMenu;

furtherMenu();

cin>>opFurtherMenu;

switch(opFurtherMenu)

{

case '1':

{

while(1)

{

system("cls");

cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;

char opFlightReservation;

flightReservationMenu();

cin>>opFlightReservation;

switch(opFlightReservation)

{

case 'a':

{

string line;

fflush(stdin);

cout<<"ENTER THE DESTINATION AIRPORT NAME: ";

getline(cin,line);

flights.searchFlight(line);

system("pause");

break;

}

case 'b':

{

int id,choice;

cout<<"ENTER FLIGHT I'D : ";

fflush(stdin);

cin>>id;

if(flights.flightExistance(id))

{

int cid;

cout<<"ENTER YOUR CUSTOMER I'D : ";

cin>>cid;

if(flights.isEmpty(cid,customer.getCustomerName(cid),id,"Bussiness"))

{

if(customer.getCustomerExistance(cid))

{

cout<<"PRESS 1 FOR BUSINESS CLASS"<<endl;

cout<<"PRESS 2 FOR ECONOMY CLASS"<<endl;

cin>>choice;

if(choice==1)

{

flights.insertPassenger(cid,customer.getCustomerName(cid),id,"Bussiness");

customer.updateCustomerTravelInfo(flights.getFlightDistance(id),cid);

}

else

{

flights.insertPassenger(cid,customer.getCustomerName(cid),id,"Economy");

customer.updateCustomerTravelInfo(flights.getFlightDistance(id),cid);

}

}

else

{

cout<<"FIRST ADD COUSTOMER IN CUSTOMER LIST"<<endl;

}

}

else

{

cout<<"Flight Is Full";

}

}

else

{

cout<<"Flight Don't Exist";

}

system("pause");

break;

}

case 'c':

{

int pid,cid;

cout<<"ENTER PLANE ID FROM WHERE YOU WANT TO DELETE PASSENGER : ";

cin>>pid;

cout<<"ENTER PASSENGER ID WHICH YOU WANT TO DELETE : ";

cin>>cid;

flights.deletePassenger(pid,cid);

customer.updateCustomerTravelInfo((flights.getFlightDistance(pid)\*(-1)),cid);

system("pause");

break;

}

case 'd':

{

system("cls");

goto furtherMenulabel;

}

default:

{

cout<<"WRONG OPTION ENTER RIGHT ONE PLEASE"<<endl;

system("pause");

}

}

system("cls");

}

break;

}

//-------------------------------Customer Menu-------------------------------------------

case '2':

{

while(1)

{

system("cls");

cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;

char opCustomerMenu;

customersMenu();

cin>>opCustomerMenu;

switch(opCustomerMenu)

{

case 'a':

{

customer.insertData();

system("pause");

break;

}

case 'b':

{

cout<<"Enter Customer I'd : ";

int num;

cin>>num;

customer.updateCustomer(num);

system("pause");

break;

}

case 'c':

{

cout<<"Enter Customer I'd You Want To Display: ";

int num;

cin>>num;

customer.searchCustomer(num);

system("pause");

break;

}

case 'd':

{

system("cls");

goto furtherMenulabel;

}

default:

{

cout<<"WRONG OPTION ENTER RIGHT ONE PLEASE"<<endl;

system("pause");

}

}

system("cls");

}

break;

}

case '3':

{

while(1)

{

system("cls");

cout<<endl<<endl<<endl<<endl<<endl<<endl<<endl<<endl;

char opFlightsMenu;

flightsMenu();

cin>>opFlightsMenu;

switch(opFlightsMenu)

{

case 'a':

{

flights.insertFlight();

system("pause");

break;

}

case 'b':

{

int upvalue;

cout<<"ENTER THE ID TO UPDATE FLIGHT : ";

cin>>upvalue;

flights.updateFlight(upvalue);

system("pause");

break;

}

case 'c':

{

int delvalue;

cout<<"ENTER THE ID TO DELETE FLIGHT : ";

cin>>delvalue;

flights.deleteFlight(delvalue);

system("pause");

break;

}

case 'd':

{

int servalue;

cout<<"ENTER THE ID TO SEARCH FLIGHT : ";

cin>>servalue;

flights.displayFlight(servalue);

system("pause");

break;

}

case 'e':

{

system("cls");

goto furtherMenulabel;

}

default:

{

cout<<"WRONG OPTION ENTER RIGHT ONE PLEASE"<<endl;

system("pause");

}

}

system("cls");

}

break;

}

case '4':

{

flights.saveData();

customer.saveData(customer.root);

system("pause");

break;

}case '5':

{

system("cls");

goto Menulabel;

}

default:

{

cout<<"WRONG OPTION ENTER RIGHT ONE PLEASE"<<endl;

system("pause");

}

}

}

break;

system("pause");

}

case '2':

{

exit(47);

}

default:

{

cout<<"WRONG OPTION ENTER RIGHT ONE PLEASE"<<endl;

system("pause");

}

}

system("cls");

}

system("pause");

return 0;

}

//-----------------------------------------------End Main--------------------------------------------