**PIMPRI CHINCHWAD EDUCATION TRUST'S**

**PIMPRI CHINCHWAD COLLEGE OF ENGINEERING**

**DEPARTMENT OF COMPUTER ENGINEERING**

****

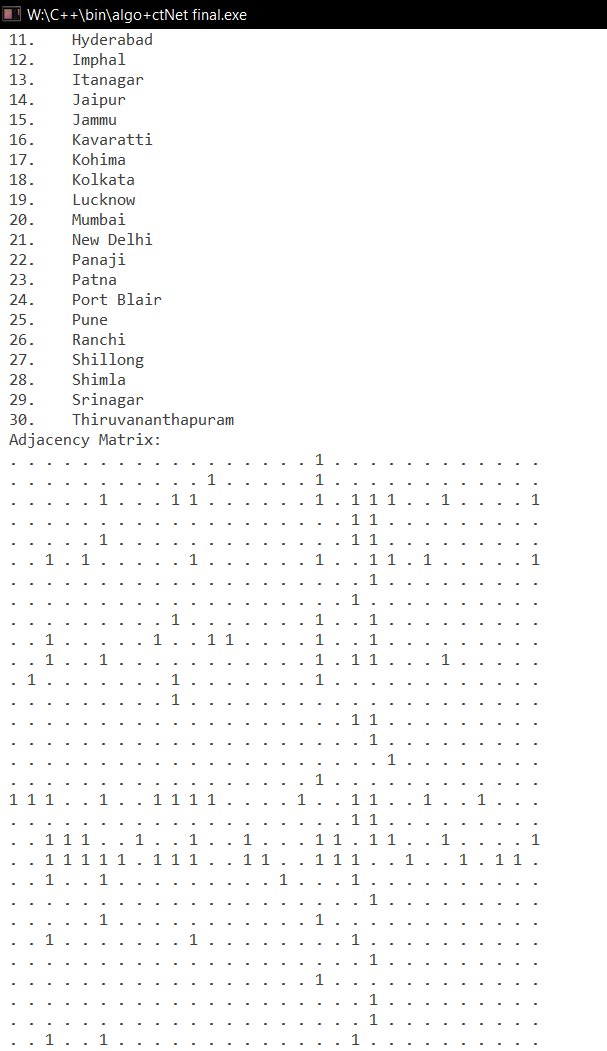
**MINI PROJECT REPORT**

**TITLE : Air India Flight booking system**

**Subject: Object Oriented Programming Lab**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Roll Number** | **Name** | **Sign** |
| **SECOB203** | Omkar Biranje |  |
| **SECOB207** | Pratik Kore |  |
| **SECOB209** | Subhradeep Mitra |  |
| **SECOB219** | Omkar Panchal |  |

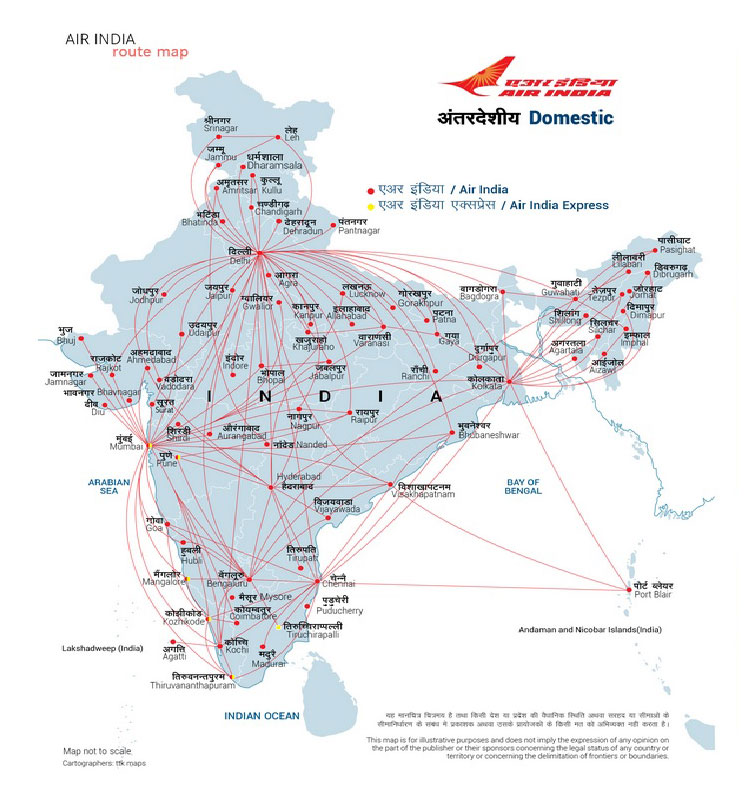
**Introduction**

Our project uses C++ language to accomplish the task of finding and booking the fastest route of connecting flights from one airport to another through a network of flights between various cities throughout India.

Features of our project include:

* Booking system that saves all records securely in binary files.
* Vast network of flights
  + 30 major cities of India
  + Real world flight times
* Finding fastest flight route
  + Achieved using Dijkstra’s Shortest Path Algorithm
  + with optimized performance using Dynamic Programming

**OOP concepts used**

1. **Data abstraction**
2. **Data encapsulation**
3. **File handling**
4. **Exception handling**
5. **Greedy approach**
6. **Dynamic programming**

PROGRAM CODE:

#include <iostream>

#include <fstream>

#include <limits.h>

#include <stack>

#include <map>

#define MAX 30

#define MAPFILE "ctlist"

#define NAMESZ 20

using namespace std;

typedef struct output\_struct

{

int total\_FlightTime;

stack<int> path;

} output;

class Map

{

int \*g[MAX][MAX]; //array of pointers

// ^simply check for existence of pointer to test connection

int n; //n= no. of vertices

map<int, output> cache;

public:

static char ctName[MAX][NAMESZ]; // search to get ct index from ct names

Map()

{

n = 0;

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

{

for (int j = 0; j < MAX; j++)

g[i][j] = NULL;

}

}

void convert(string str = "")

{

if (!str.size())

{

cout << "\n Open (text file name): ";

cin >> str;

}

str += ".txt";

fstream db;

db.open(str.c\_str(), ios::in);

n = 0;

string ct;

// read city names until line "." occurs

getline(db, ct);

while (db && ct != ".")

{

ctName[n][ct.copy(ctName[n], ct.size(), 0)] = '\0';

getline(db, ct);

n++;

}

ct = "";

int x, y;

//read edge coordinates, cost, time(minutes) on each line

while (db >> x >> y) //natural no.s

{

x--;

y--;

g[x][y] = new int;

db >> \*g[x][y];

g[y][x] = g[x][y];

}

db.close();

// start writing

int i = 3;

/\*while (i--)

str.pop\_back();\*/

str += "bin";

db.open(str.c\_str(), ios::out);

db.write((char \*)&n, sizeof(n)); // no. of cts

db.write((char \*)&ctName, (n + 1) \* NAMESZ); //ctName truncated to NAMESZ bytes(chars)

int \*no = new int;

\*no = 0;

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

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

{

if (!g[i][j])

g[i][j] = no; //if cost/time =0 => not connected

db.write((char \*)g[i][j], sizeof(int));

}

db.close();

}

void load(string str = "")

{

if (!str.size())

{

cout << "\n Open (binary file name):";

cin >> str;

}

str += ".bin";

ifstream db(str.c\_str(), ios::binary);

db.read((char \*)&n, sizeof(n)); // read no. of cts

db.read((char \*)&ctName, (n + 1) \* NAMESZ); // read ct names

ctName[n][1] = '\0';

int e = 0; //if time =0 => not connected

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

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

{

db.read((char \*)&e, sizeof(int)); //read into e

if (!e) // check if connected

g[i][j] = g[j][i] = NULL; //null pointer if not

else

{

g[i][j] = new int; // new edge if yes

\*g[i][j] = e;

g[j][i] = g[i][j];

}

}

db.close();

}

void sho()

{

cout << "\n No.(Cities):" << n;

cout << "\n City names:";

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

cout << "\n " << i + 1 << ".\t" << ctName[i];

cout << "\n Adjacency Matrix:";

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

{

cout << endl;

for (int j = 0; j < n; j++)

cout << " " << (g[i][j] ? "1" : ".");

}

cout << "\n Raw edge data:";

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

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

if (g[i][j])

cout << endl

<< i + 1 << " " << j + 1 << " " << \*g[i][j];

}

output skip(int i, int j) // Dynamic programming check

{

int t;

output o;

try

{

o = cache.at(MAX \* min(i, j) + max(i, j)); // will it work?

if (i > j)

{

stack<int> s = o.path;

while (!o.path.empty())

o.path.pop();

while (!s.empty())

{

o.path.push(s.top());

s.pop();

}

cout << "\n Skipped from " << ctName[i] << " to " << ctName[j] << endl;

return o;

}

}

catch (exception &out\_of\_range)

{

o.total\_FlightTime = 0;

return o;

}

}

output dijkstra(int source, int destination)

{

output o;

o.total\_FlightTime = 0;

o = skip(source, destination);

if (o.total\_FlightTime)

return o;

int prev\_Node[MAX], label[MAX], mini, min\_index, source1, destination1;

bool visited[MAX];

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

prev\_Node[i] = source, label[i] = INT\_MAX, visited[i] = false;

label[source] = 0, visited[source] = true;

source1 = source;

destination1 = destination;

while (source != destination)

{

mini = INT\_MAX;

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

if (g[source][i] && !visited[i] && label[i] >= label[source] + \*g[source][i])

{

label[i] = label[source] + \*g[source][i];

prev\_Node[i] = source;

}

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

if (!visited[i] && mini >= label[i])

{

mini = label[i];

min\_index = i;

}

o = skip(min\_index, destination1);

if (o.total\_FlightTime)

break;

visited[min\_index] = true;

source = min\_index;

}

o.total\_FlightTime += label[source];

while (source != source1)

{

o.path.push(source);

source = prev\_Node[source];

}

o.path.push(source);

cache.insert(pair<int, output>(MAX \* min(source1, destination1) + max(source1, destination1), o));

return o;

}

};

char Map::ctName[MAX][NAMESZ];

typedef struct ticket

{

char name[NAMESZ], from[NAMESZ], to[NAMESZ];

int n, id, day, month, year, f, t, cost, time;

} ticket;

class Flight

{

int row[3], countp, countf, np, s, d;

char no;

int tseat[MAX];

char tno[MAX];

int cost;

stack<int> path;

ticket t;

Map ctNet;

public:

Flight()

{

ctNet.load(MAPFILE);

row[0] = 1, row[1] = 0, row[2] = 0, countp = 0;

no = '@';

t.id = 0;

}

void start()

{

while (1)

{

cout << "\nEnter today's DATE : \nDAY : ";

cin >> t.day;

cout << "\nMonth : ";

cin >> t.month;

cout << "\nYear : ";

cin >> t.year;

if ((t.day >= 31 && (t.month == 4 || t.month == 6 || t.month == 9 || t.month == 11)) || (t.day >= 32 && (t.month == 1 || t.month == 3 || t.month == 5 || t.month == 7 || t.month == 8 || t.month == 10)) || (t.day >= 30 && t.month == 2 && t.year % 4 == 0) || (t.day >= 29 && t.month == 2 && t.year % 4 != 0))

{

cout << "\nINVALID DATE...Please enter correct date.\n";

continue;

}

else

break;

}

if (t.day == 31 && t.month == 12)

{

t.day = 1;

t.month = 1;

t.year++;

}

else if ((t.day == 30 && (t.month == 4 || t.month == 6 || t.month == 9 || t.month == 11)) || (t.day == 31 && (t.month == 1 || t.month == 3 || t.month == 5 || t.month == 7 || t.month == 8 || t.month == 10)))

{

t.day = 1;

t.month++;

}

else if ((t.day == 29 && t.month == 2 && t.year % 4 == 0) || (t.day == 28 && t.month == 2))

{

t.day = 1;

t.month++;

}

else

{

t.day++;

}

}

int cost\_cal(output o, int p)

{

path = o.path;

t.time = o.total\_FlightTime;

return p \* ((o.total\_FlightTime \* 12) / 4);

}

int source()

{

while (1)

{

cout << "\nPlease choose one from the list of the cities mentioned below : ";

int i;

for (i = 0; i < MAX && ctNet.ctName[i][1]; i++)

{

cout << "\n"

<< i + 1 << ". " << ctNet.ctName[i];

}

cout << "\n\nEnter SOURCE NO : ";

cin >> s;

try

{

if (s > i)

throw s;

break;

}

catch (int)

{

cout << "\nINVALID CHOICE OF INDEX.Please repeat the process.";

continue;

}

}

return s;

}

int destination()

{

int i;

while (1)

{

cout << "\nPlease choose one from the list of the cities mentioned below : ";

for (i = 0; i < MAX && ctNet.ctName[i][1]; i++)

{

if (s - 1 == i)

{

break;

}

else

cout << "\n"

<< i + 1 << ". " << ctNet.ctName[i];

}

i++;

for (; i < MAX && ctNet.ctName[i][1]; i++)

cout << "\n"

<< i + 1 << ". " << ctNet.ctName[i];

cout << "\n\nEnter DESTINATION NO : ";

cin >> d;

try

{

if (d > i)

throw d;

break;

}

catch (int z)

{

cout << "\nINVALID CHOICE OF INDEX.Please repeat the process.";

continue;

}

}

return d;

}

void book(bool enq = 0)

{

cout << "\nEnter the number of passengers : ";

cin >> np;

try

{

if (np > 7)

throw np;

countp = countp + np;

if (countp <= 50)

{

if (np == 1)

cout << "\nEnter Name of Passenger : ";

else

cout << "\nEnter name of the HEAD of your group : ";

cin >> t.name;

t.id++;

t.n = np;

t.f = source() - 1;

t.t = destination() - 1;

t.cost = cost\_cal(ctNet.dijkstra(t.f, t.t), np);

for (int i = 0; i < t.n; i++)

{

seat();

tseat[i] = row[1] \* 10 + row[0];

tno[i] = no;

}

if (!enq)

{

ofstream file;

file.open("BOOKINGS.DAT", ios::app);

file.write((char \*)&t, sizeof(t));

file.close();

cout << "\n Your ticket was booked successfully!";

}

else

cout << "\n This a provisional ticket:";

issue\_ticket();

}

else

cout << "\nFlight is FULL !!! Please try again tommorow.\n";

}

catch (int a)

{

cout << "\nEXCEPTION CAUGHT!!!";

cout << "\nYOU CANNOT BOOK MORE THAN 7 TICKETS\n";

}

}

void issue\_ticket()

{

cout << "\n\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\t| |";

cout << "\n\t|\tAIR INDIA\t\t\t\t\t\t\t\t\t\t"

<< "ID : " << t.id << "\t |"

<< "\n\t| |";

cout << "\n\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\t| |";

cout << "\n\t|\tPassenger Name : " << t.name << "\t\t\t\t\t\tNumber of seats : " << t.n<<"\t |";

if (np > 1)

cout << "\n\t| |\n\t|\tFrom : " << ctNet.ctName[t.f] << "\t\t\t\t\t\t\t\tSeat No : " << tseat[0] << tno[0] << " to " << tseat[t.n - 1] << tno[t.n - 1] << "\t |";

else

cout << "\n\t| |\n\t|\tFrom : " << ctNet.ctName[t.f] << "\t\t\t\t\t\t\t\tSeat No : " << tseat[0] << tno[0] << "\t\t |";

cout << "\n\t| |\n\t|\tTo : " << ctNet.ctName[t.t] << "\t\t\t\t\t\t\t\tDate : " << t.day << "/" << t.month << "/" << t.year << "\t |\n\t| |";

cout << "\n\t|\tCost : Rs. " << t.cost << "/-\t\t\t\t\t\t\tDuration : " << t.time / 60 << "hrs " << t.time % 60 << "min"

<< "\t |\n\t| |\n\t| |";

cout << "\n\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n\t| |";

cout << "\n\t|\t\t\t\t\t\tHAPPY JOURNEY\t\t\t\t\t\t |\n\t| |";

cout << "\n\t|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|\n";

cout << "\n Your course will be\n";

while (!path.empty())

{

cout <<" "<< ctNet.ctName[path.top()];

if (path.size() > 1)

cout << " -> ";

path.pop();

}

cout << endl;

}

void seat()

{

if (row[1] < 1)

{

int t = (int)no;

if (t < 69)

{

t++;

no = (char)t;

}

else if (t == 69)

{

no = 'A';

if (row[0] == 9)

{

if (row[1] < 1)

{

row[1]++;

}

}

else

row[0]++;

}

}

}

void display()

{

cout << "ID : " << t.id;

cout << "\nName of Passenger : " << t.name;

cout << "\nNo of seats : " << t.n;

cout << "\nCost : " << t.cost << endl;

}

void database()

{

long int pass;

//= 37919;

int j = 5, flag = 0, i;

char p;

do

{

if (j < 5)

{

cout << "\n"

<< j << " chances left.\n";

cout << "\nForgot password ??? (y/n) : ";

cin >> p;

if (p == 'y')

{

cout << "\nSECURITY QUESTION - ENTER YOUR FAVOURITE NUMBER : ";

cin >> pass;

}

else

{

cout << "\nEnter ADMIN password : ";

cin >> pass;

}

}

else

{

cout << "\nEnter ADMIN password : ";

cin >> pass;

}

ifstream d;

switch (pass)

{

case 7:

case 37919:

i = 1;

d.open("BOOKINGS.DAT", ios::in);

while (d)

{

d.read((char \*)&t, sizeof(t));

if (d.eof())

break;

cout << "\n"

<< i << ". ";

display();

i++;

}

d.close();

flag = 1;

break;

default:

j--;

cout << "\nINCORRECT PASSWORD\n";

}

} while (flag == 0 && j > 0);

if (j == 0)

cout << "\n";

}

};

int main()

{

Flight f;

f.start();

char ch;

bool check = 0;

do

{

cout << "\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~WELCOME to AIR\_INDIA~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n"

<< endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\* MENU \*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Enter\n1 - Book New Ticket" << endl;

cout << "2 - Enquiry" << endl;

cout << "3 - Show DATABASE (only for administrators)" << endl;

cout << "e - Exit.\n";

cin >> ch;

switch (ch)

{

case '1':

f.book();

break;

case '2':

f.book(1);

break;

case '3':

f.database();

break;

case 'e':

check = 1;

break;

default:

cout << "\nINVALID ENTRY!!!" << endl;

break;

}

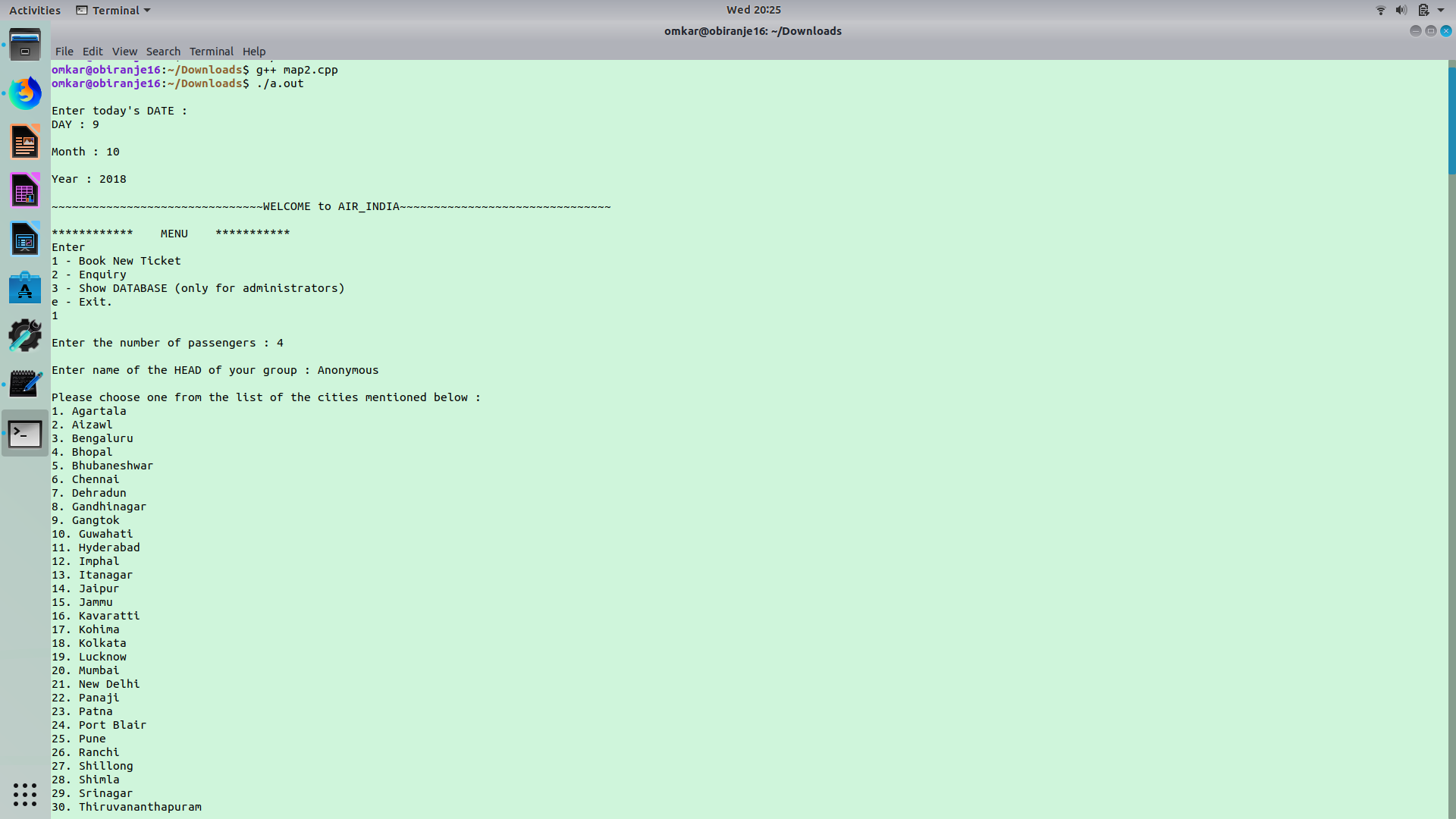
} while (check == 0);

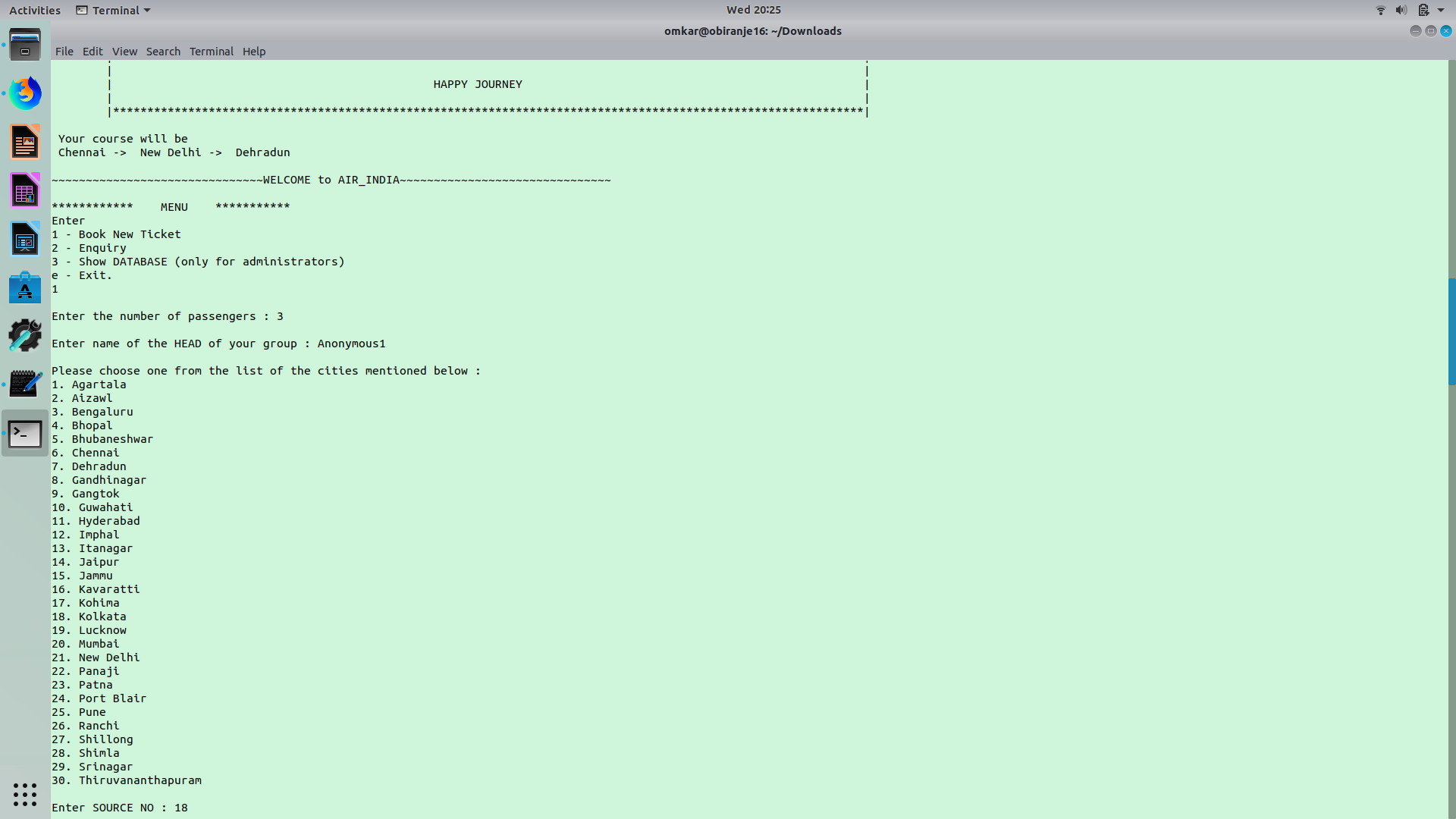
cout << "\nTHANK YOU!!!\n";

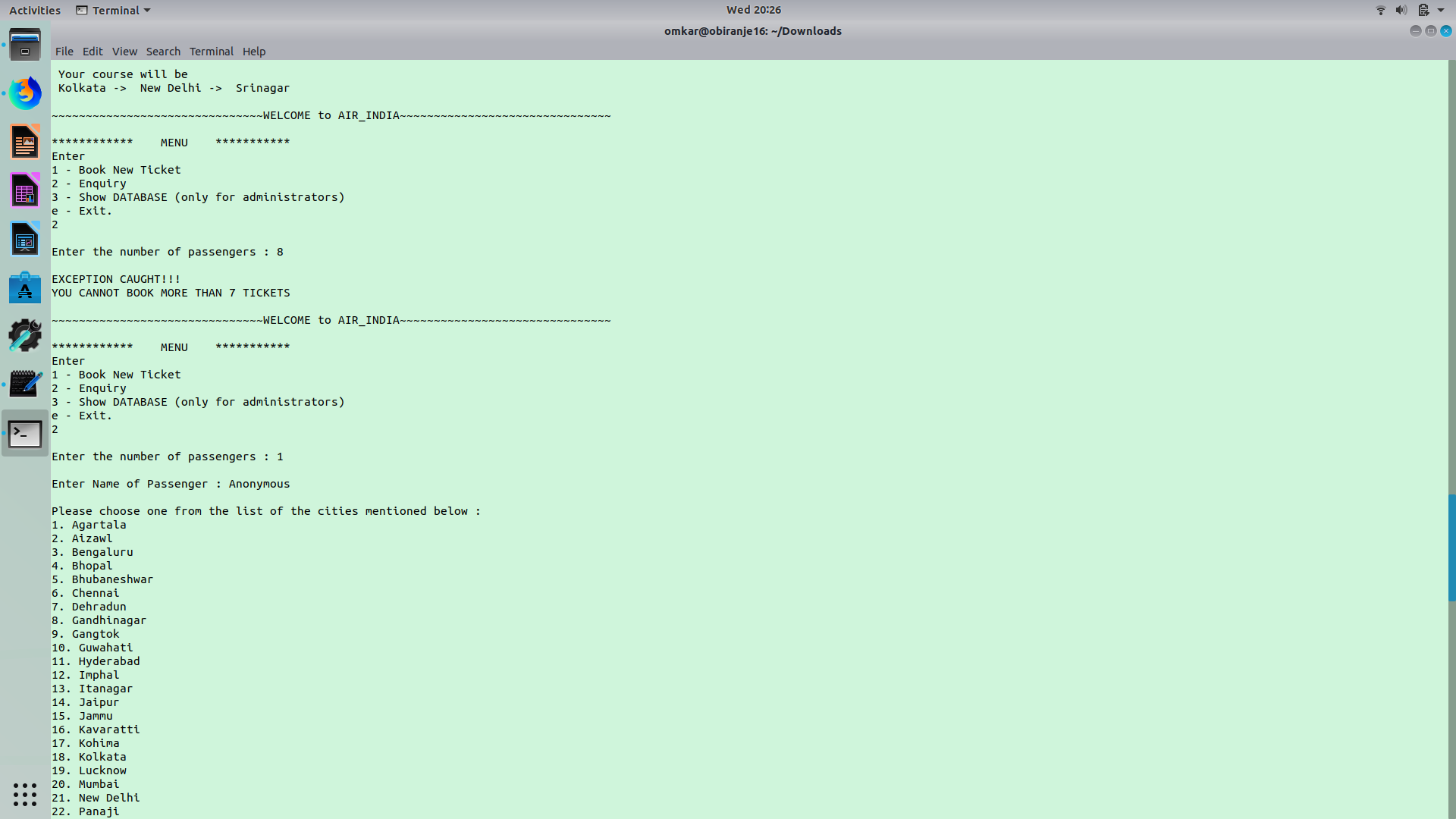
return 0;

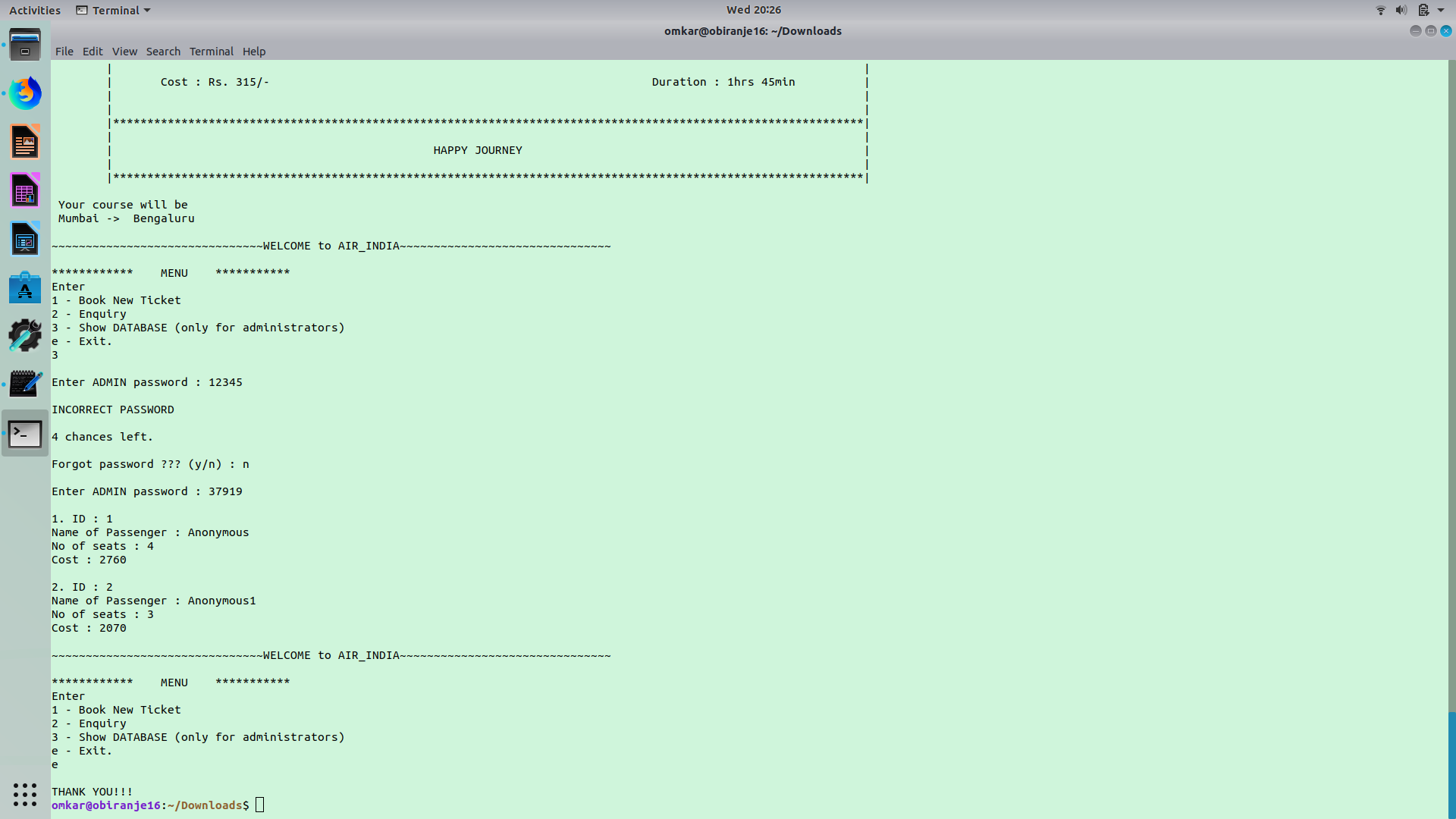
}

OUTPUT



****

****

****

**Sign of Faculty**

**(Mr. Anand Birajdar)**