**1. One paragraph description of the project.**

Given a list of 18 airports, flight distances, cost and time for travel between the airports, the project was to come up with a program that assisted travellers with navigating through the travel options to find the best choices for the shortest distance, cheapest cost and fastest time from departure point to destination.

**2. One paragraph description of the data structures used and why you selected this data structure(s) for the implementation.**

We used a graph: a graph has vertexes and edges. In our case these were airports and flights. There are 3 classes to represent these:

Graph - Holds all the vertexes

Airport - A vertex with edges to the other vertexes

Flight – Edges

This data structure was used because it mirrors the problem. There is a 1-to-1 correspondence between the graph and the classes used.

**3. A step by step instruction of how to use the code. This should include a test case for running your code.**

Please compile the code and run it as a console application.

The following screen appears when the program is executed – the list of 18 airports and two input prompts.

1: JFK

2: SFO

3: BWI

4: DEN

5: LAX

6: ORD

7: IAD

8: DFW

9: BOS

10: SEA

11: ATL

12: DCA

13: IND

14: IAH

15: MIA

16: TPA

17: HSV

18: JAN

Enter Departure Airport:

Enter Destination Airport:

a. Follow screen prompt to input the Departure Airport from the list provided (not case sensitive). E.g. “Den”: Enter Departure Airport: Den

b. Follow screen prompt to input the Destination Airport from the list provided (not case sensitive). E.g. “HSV”: Enter Destination Airport: HSV

c. The program will give the following output:

Cheapest: $1138.1

DEN to ATL $1079.1 leave 1234 arrive 1527 (173 minutes) 1199 miles

layover: 15 hours 36 minutes.

ATL to HSV $59 leave 703 arrive 752 (49 minutes) 151 miles

Shortest: 1350 miles

DEN to ATL $1079.1 leave 1234 arrive 1527 (173 minutes) 1199 miles

layover: 15 hours 36 minutes.

ATL to HSV $59 leave 703 arrive 752 (49 minutes) 151 miles

Fastest: 737 minutes

DEN to ATL $1079.1 leave 1935 arrive 2220 (165 minutes) 1199 miles

layover: 8 hours 43 minutes.

ATL to HSV $59 leave 703 arrive 752 (49 minutes) 151 miles

Press any key to continue . .

**4. List the IDE(s) and OS(s) used during development and testing.**

We used Microsoft Visual Studio Versions 10, 13 and 15 for initial development and testing, but only Microsoft Visual Studio Versions 13 and 15 for final development and testing.

**5. If you worked with a partner, your documentation must also include a listing of who did what.**

We had a unique approach to the project, where we worked separately using different approaches and then combined our code at the end. My implementation ran faster, so we chose to submit it. However, with my program, I had a problem writing the code to split the string when reading in the csv file. I foraged through books and the Internet and found code – written in C – that worked, but only in Microsoft Visual Studio Versions 10 and 15 (I had access to Versions 10 and 13; Pawel worked with Version 15). (see code below) The following additional file headers were required for the program to run : #define \_CRT\_SECURE\_NO\_WARNINGS; #include <cstdlib> // atoi, strtod; #include <cstring> // strtok.

// skip the header

getline(file, line);

// process each flight in the CSV file

while (getline(file, line))

{

#ifdef \_WIN32

char \* p;

string depart = strtok\_s((char\*)line.c\_str(), ",", &p);

string arrive = strtok\_s(NULL, ",", &p);

int timeLeave = atoi(strtok\_s(NULL, ",", &p));

int timeArrive = atoi(strtok\_s(NULL, ",", &p));

int distance = atoi(strtok\_s(NULL, ",", &p));

double price = strtod(strtok\_s(NULL, "\r\n", &p), NULL);

#else

string depart = strtok((char\*)line.c\_str(), ",");

string arrive = strtok(NULL, ",");

int timeLeave = atoi(strtok(NULL, ","));

int timeArrive = atoi(strtok(NULL, ","));

int distance = atoi(strtok(NULL, ","));

double price = strtod(strtok(NULL, "\r\n"), NULL);

#endif

airport\_graph.add\_flight(depart, arrive, timeLeave, timeArrive,

distance, price);

}

file.close();

In the submitted program, we replaced this portion of code with code that Pawel wrote for his program – his program can be accessed through the following link: <https://github.com/pgongit/CS271_Final/blob/master/PuddleJumper.cpp>. His code finds the positions of the comma's in the line of the csv file and then takes the part of the string between the commas and copies a part of the string from the positions given.

// Skip the header

getline(file, flights);

// process each flight in the CSV file

while (file.peek() != EOF)

{

file >> flights;

int pos = 0;

int count = 0;

string depart, arrive = "";

int timeLeave, timeArrive, distance = 0;

double price = 0;

int delimiterPos[5];

for (int i = 0; i < flights.length(); i++)

{

if (flights[i] == ',')

{

delimiterPos[count++] = pos;

}

pos++;

}

depart = flights.substr(0, delimiterPos[0]);

arrive = flights.substr(delimiterPos[0] + 1, (delimiterPos[1] - delimiterPos[0] - 1));

timeLeave = stoi(flights.substr(delimiterPos[1] + 1, (delimiterPos[2] - delimiterPos[1] - 1)));

timeArrive = stoi(flights.substr(delimiterPos[2] + 1, (delimiterPos[3] - delimiterPos[2] - 1)));

distance = stoi(flights.substr(delimiterPos[3] + 1, (delimiterPos[4] - delimiterPos[3] - 1)));

price = stod(flights.substr(delimiterPos[4] + 1, (delimiterPos[5] - delimiterPos[4] - 1)));

airport\_graph.add\_flight(depart, arrive, timeLeave, timeArrive, distance, price);

}

file.close();