CSCE 221
Assignment # 6

First Name: Jialu Last Name: Zhao UIN: 426000712

User Name: zhaojialu123 E-mail address: zhaojialu123@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People					
Web pages (provide URL)					
Printed material					
Other Sources					

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Jialu Zhao Date 04/27/2017

1.The description of an assignment problem.

Part1

Parse the input file (input1.data, input2.data, input3.data) to obtain the departure and destination city for each flight. The cities names are integers. Each line (starting from line 2) in the input file contains destination cities accessible directly from a departure city. The departure city is given implicitly, and is equal to the line number minus 2 (so it starts from 0). Numbers listed in any line corresponds only to the destination cities, and they are in the following format:

destination_city_1 destination_city_2 ... -1

(-1 is used as an end-of-line sentry) which means that there are direct flights from the departure city to any other destination city in the line. The first line contains two numbers: the number of departure cities (equal to the number of all lines but the first one), and the number of direct flights.

Create a graph where each city is a vertex and a flight between two cities is the edge between them. A graph is represented by an adjacency list structure. Notice that each node in a linked list corresponds to the direct flight from the departure city.

Part2

In the part 2 of the assignment use this graph data structure and determine the possibility of partition the cities into two groups such that there is no edge between two vertices in each group. The algorithm should output two groups of cities, or issue a message that such a grouping is not possible. The groups should be disjoint, that is, they do not have a common city. Use the provided input files for testing.

Part3

In the part 3 of the assignment you need to find the shortest path (minimum of interim stops) for two given vertices (cities) in the same group, and list all the vertices (cities) on this shortest path.

2.The description of data structures and algorithms used to solve the problem.

(a)Provide definitions of data structures by using Abstract Data Types (ADTs)

(1) Graph:

A graph G is a pair (V,E), where V is a set of vertices, and E is a set of edge. Each edgeis associated with two vertices,which are endpoints of the edge.

(2) Priority Queue:

A priority queue P is a data type that consists of prioritized entries. An entry is a pair (key,value). The key represents the priority which does not need to be unique for two distinct entries.

ADT:

insertItem(k, x) inserts a key k and value x into P.

minElement() returns an entry with the smallest key in P (but it does not remove it).

removeMin() removes an entry with the smallest key in P.

minKey() returns the smallest key in P.

isEmpty() tests if P is empty

(b)Write about the ADTs implementation in C++.

(1) Graph:

```
Graph(){} // default constructor, implementation doing nothing
Graph(vector<list<Edge>> adjl); // constructor from adjl = adjacency list (optional)
void buildGraph(ifstream &in); // build a graph from the adjacency list
void displayGraph(ostream &cout); // display the graph
void determine_partition(); // determine the possibility of partition the cities into two groups
void find_shortest_path(Vertex v1, Vertex v2, Graph g); // find the shortest path
```

(2) Priority Queue

```
priority_queue<Vertex,vector<Vertex>, compare > PQ; // declare priority queue
struct compare{};// using for defintion of priority queue
pop(); // extract the first element of the priority queue
push(); // push a new element to the priority queue
top(); // get the first element of the priority queue
empty(); // judge if the priority queue is empty or not
```

(c)Describe algorithms used to solve the problem.
Algorithm:

```
step1: Parse the input file to obtain the start and end vertices
step2: Create a graph where each city is a vertex and a flight between two cities is an edge betwee
step3: Use several for loops to determine if there exists grouping can seperate cities to two group
step4: Find the shortest path between two vertices in the same group
```

3.Provide features of the C++ programming paradigms like Inheritance or Polymorphism in case of object oriented programming, or Templates in the case of generic programming used in your implementation.

(1) Vertex class, Edge class, Graph class for interface
(2) Struct tags
(3) const using
(4) reference using

4.A user guide description how to navigate your program with the instructions how to:

(a)compile the program: specify the directory and file names, etc.

Type: make in the command window

(b)run the program: specify the name of an executable file.

Type:./main+input.data in the command window

5.Specifications and description of input and output formats and files

(a)The type of files: keyboard, text files, etc (if applicable).

(1) Input: txt file

The input file (input1.data, input2.data, input3.data) to obtain the departure and destination city for each flight. The cities names are integers. Each line (starting from line 2) in the input file contains destination cities accessible directly from a departure city. The departure city is given implicitly, and is equal to the line number minus 2 (so it starts from 0). Numbers listed in any line corresponds only to the destination cities, and they are in the following format:

destination_city_1 destination_city_2 ... -1

(-1 is used as an end-of-line sentry) which means that there are direct flights from the departure city to any other destination city in the line. The first line contains two numbers: the number of departure cities (equal to the number of all lines but the first one), and the number of direct flights.

(2) Output:

Part1:graph, using numbers only

Departure_city_1: Destination_city Destination_city ...
Departure_city_2: Destination_city Destination_city ...
Departure_city_3: Destination_city Destination_city ...

Part2:

(1) If there exists this grouping:

Group1: 0 3 4
Group2: 1 2 5 6

(2) If it's impossible for this grouping:

output:such a grouping is not possible.

Part3:

(1) If there exists and the vertices users choose in the same group:

Please choose two cities which you want to find the shortest path between them:
(Note: Choose from the same group, using space to seperate them!!)
0 2
The shortest path from 0 to 2 is 2.
The number of stops is: 3.
All the vertices (cities) on this shortest path: 0->1->2

(2) If it's impossible for this grouping:

This grouping is not existed, so can't find the shortest path from cities.

(3) If the two vertices that users choose are not in the same group or either of them doesn't exist in these two groups:

Please choose two cities which you want to find the shortest path between them: (Note: Choose from the same group!!)
0 1
These two vertices are not in the same group or they don't exist in these groups!!

(b)A file input format: when a program requires a sequence of input items, specify the number of items per line or a line termination. Provide a sample of a required input format.

When asking to input start and end city, type two intergers in the command window.

Note: Using whitespace to seperate them!!

(c)Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

The users are supposed to input two intergers and use whitespace to seperate them, if the user didn't use space to seperate them, it will crash.

6.Provide exceptions and their purpose in your program.

(1) If it's impossible for this grouping, it will throw an exception.

(2) If the two vertices that users choose are not in the same group or either of them doesn't exist in these two groups, it will throw an exception.

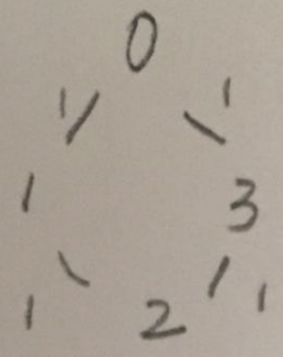
Both of them are run time exception.

7.Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output.

(1) Using input1.data

(a) Test by hand:

input 1. data



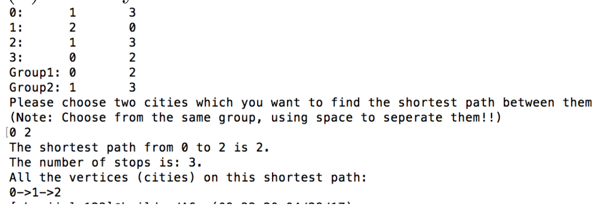
group 1: 0 2

group 2 : 1 3

shortest path from 0 to 2 is 1+1=

input 2. data

(b) Test by code:



```
0:      1       3
1:      2       0
2:      1       3
3:      0       2
Group1: 0       2
Group2: 1       3
Please choose two cities which you want to find the shortest path between them
(Note: Choose from the same group, using space to seperate them!!)
0 2
The shortest path from 0 to 2 is 2.
The number of stops is: 3.
All the vertices (cities) on this shortest path:
0->1->2
```

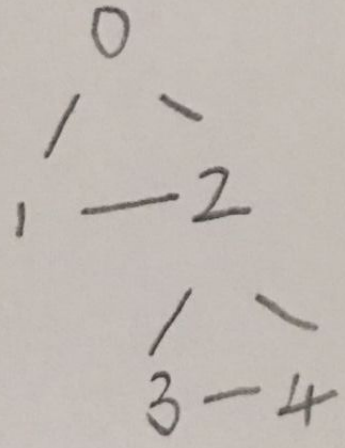
(2)using input2.data to test (test this grouping is impossible)

(a)test by hand

group2 : 1 3

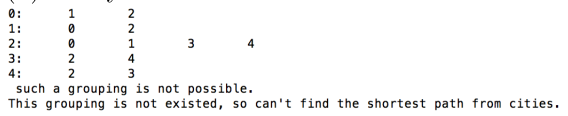
shortest path from 0 to 2 is

input 2. data



Such grouping is impossible.

(b)test by code



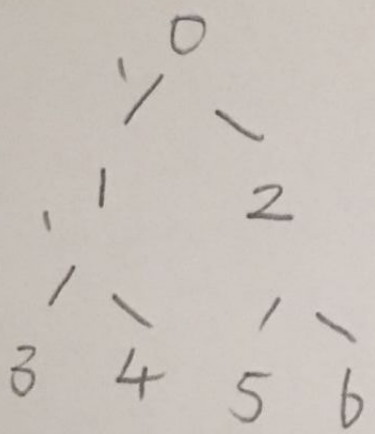
```
0:      1       2
1:      0       2
2:      0       1       3       4
3:      2       4
4:      2       3
 such a grouping is not possible.
This grouping is not existed, so can't find the shortest path from cities.
```

(3) using input3.data to test (test invaild input: not in the same group)

(a) test by hand

Such grouping is impossible.

input 3. data

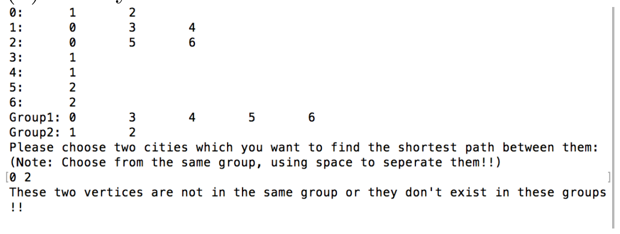


group 1 : 0 3 4 5 6

group 2 : 1 2

shortect path from 0 to 3 = 1+1 =

(b) test by code

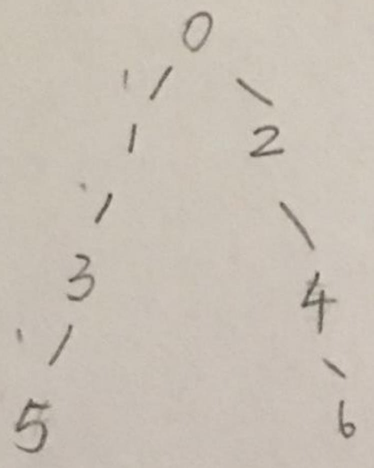


```
0:      1       2
1:      0       3       4
2:      0       5       6
3:      1
4:      1
5:      2
6:      2
Group1: 0       3       4       5       6
Group2: 1       2
Please choose two cities which you want to find the shortest path between them:
(Note: Choose from the same group, using space to seperate them!!)
0 2
These two vertices are not in the same group or they don't exist in these groups
!!
```

(4) using input4.txt to test(create by myself)

(a) test by hand

input 4. data

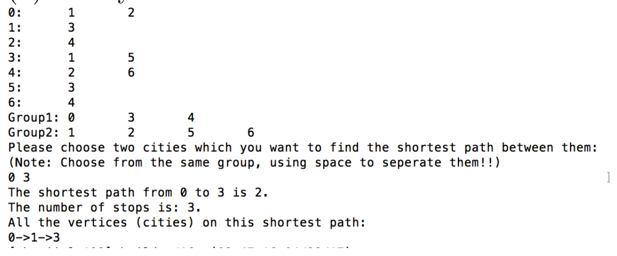


group 1 : 0 3 4

group 2 : 1 2 5 6

shortest path from 0 to 3 is 1+1 =

(b) test by code:



```
0:      1       2
1:      3
2:      4
3:      1       5
4:      2       6
5:      3
6:      4
Group1: 0       3       4
Group2: 1       2       5       6
Please choose two cities which you want to find the shortest path between them:
(Note: Choose from the same group, using space to seperate them!!)
0 3
The shortest path from 0 to 3 is 2.
The number of stops is: 3.
All the vertices (cities) on this shortest path:
0->1->3
```