**GROUP C**

**There are flight paths between cities. If there is a flight between city A and city B then**

**there is an edge between the cities. The cost of the edge can be the time that flight take**

**to reach city B from A, or the amount of fuel used for the journey. Represent this as a**

**graph. The node can be represented by airport name or name of the city. Use adjacency**

**list representation of the graph or use adjacency matrix representation of the graph.**

**Check whether the graph is connected or not. Justify the storage representation used**

#include <iostream>

#include <queue>

using namespace std;

int adj\_mat[50][50] = {0, 0};

int visited[50] = {0};

void dfs(int s, int n, string arr[])

{

visited[s] = 1;

cout << arr[s] << " ";

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

{

if (adj\_mat[s][i] && !visited[i])

dfs(i, n, arr);

}

}

void bfs(int s, int n, string arr[])

{

bool visited[n];

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

visited[i] = false;

int v;

queue<int> bfsq;

if (!visited[s])

{

cout << arr[s] << " ";

bfsq.push(s);

visited[s] = true;

while (!bfsq.empty())

{

v = bfsq.front();

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

{

if (adj\_mat[v][i] && !visited[i])

{

cout << arr[i] << " ";

visited[i] = true;

bfsq.push(i);

}

}

bfsq.pop();

}

}

}

int main()

{

cout << "Enter no. of cities: ";

int n, u;

cin >> n;

string cities[n];

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

{

cout << "Enter city #" << i << " (Airport Code): ";

cin >> cities[i];

}

cout << "\nYour cities are: " << endl;

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

cout << "city #" << i << ": " << cities[i] << endl;

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

{

for (int j = i + 1; j < n; j++)

{

cout << "Enter distance between " << cities[i] << " and " << cities[j] << " : ";

cin >> adj\_mat[i][j];

adj\_mat[j][i] = adj\_mat[i][j];

}

}

cout << endl;

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

cout << "\t" << cities[i] << "\t";

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

{

cout << "\n"

<< cities[i];

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

cout << "\t" << adj\_mat[i][j] << "\t";

cout << endl;

}

cout << "Enter Starting Vertex: ";

cin >> u;

cout << "DFS: ";

dfs(u, n, cities);

cout << endl;

cout << "BFS: ";

bfs(u, n, cities);

return 0;

}

**OUTPUT:**

**Enter no. of cities: 2**

**Enter city #0 (Airport Code): 845**

**Enter city #1 (Airport Code): 454**

**Your cities are:**

**city #0: 845**

**city #1: 454**

**Enter distance between 845 and 454 : 555**

**845 454**

**845 0 555**

**454 555 0**

**Enter Starting Vertex: 555**

**Segmentation fault**