```
#include <iostream>
#include <string>
#include <stack>
#include <queue>
using namespace std;
void DFS(int vertex, int n, int cost[10][10], string cities[], bool visited[])
{
cout << cities[vertex] << " ";
visited[vertex] = true;
for (int i = 0; i < n; ++i) {
if (cost[vertex][i] != 0 && !visited[i]) {
DFS(i, n, cost, cities, visited);
}
}
}
void BFS(int vertex, int n, int cost[10][10], string cities[]) {
bool visited[10] = {false};
queue<int> q;
cout<<endl;
cout<< "BFS Traversal : "<<endl;
q.push(vertex);
visited[vertex] = true;
while (!q.empty()) {
int current = q.front();
q.pop();
cout << cities[current] << " ";
for (int i = 0; i < n; i++) {
if (cost[current][i] != 0 && !visited[i]) {
q.push(i);
visited[i] = true;
}
}
}
cout << endl;
}
void graph(int n, int cost[10][10]) {
cout << "\n\nAdjacency Matrix representation of graph:" << endl;
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++) {
cout << cost[i][j] << "\t";
}
cout << endl;
}
}
void display(int n, string cities[]) {
cout << "Display Cities" << endl;for (int i = 0; i < n; i++) {
cout << "city" << i + 1 << ": " << cities[i] << endl;
}
}
int main() {
int cost[10][10] = {0};
int n, i, j;
char op;
int vertex;
cout << "enter total number of cities: ";
cin >> n;
string cities[n];
for (i = 0; i < n; i++) {
cout << "Enter city " << i + 1 << ": ";
cin >> cities[i];
}
for (i = 0; i < n; i++) {
for (j = i + 1; j < n; j++) {
cout << "Is there edge between " << cities[i] << " and " << cities[j] << " (y/n)? ";
cin >> op;
if (op == 'y' || op == 'Y') {
cout << "Enter cost: ";
cin >> cost[i][j];
cost[j][i] = cost[i][j];
}
}
}
display(n, cities);
graph(n, cost);
cout << "\nenter the starting vertex for traversal: ";
cin >> vertex;
bool visited[10] = {false};
cout << "DFS Traversal : ";
cout<<endl;
DFS(vertex, n, cost, cities, visited);
BFS(vertex, n, cost, cities);
return 0;
}
```