

Assignment no 6

/*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 takes 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 MATRIX representation of the graph.*/

```
#include<iostream>
```

```
#include<queue>
```

```
#include<stack>
```

```
using namespace std;
```

```
class Graph {
```

```
    string city[10];
```

```
    int a[10][10];
```

```
    int n;
```

```
public:
```

```
    void input();
```

```
    void display();
```

```
    void BFS();
```

```
    void DFS();
```

```
};
```

```
void Graph::input() {
```

```
    cout << "\nEnter number of cities: ";
```

```
    cin >> n;
```

```
    cout << "\nEnter names of cities:\n";
```

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

```
        cout << "City " << i + 1 << ": ";
```

```

        cin >> city[i];
    }

    cout << "\nEnter distances between cities:\n";

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (i == j) {
                a[i][j] = 0;
            } else {
                cout << "Distance from " << city[i] << " to " << city[j] << ": ";
                cin >> a[i][j];
            }
        }
    }
}

```

```

void Graph::display() {
    cout << "\nAdjacency Matrix:\n\t";

    for (int i = 0; i < n; i++) {
        cout << city[i] << "\t";
    }

    cout << endl;

    for (int i = 0; i < n; i++) {
        cout << city[i] << "\t";

        for (int j = 0; j < n; j++) {
            cout << a[i][j] << "\t";
        }

        cout << endl;
    }
}

```

```

void Graph::BFS() {

    cout << "\n\nBFS Traversal:\n";

    queue<int> q;

    int visit[10] = {0};

    string start;

    int index = -1;


    cout << "Enter starting city: ";

    cin >> start;

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

        if (start == city[i]) {

            index = i;

            break;

        }

    }

    if (index == -1) {

        cout << "City not found.\n";

        return;

    }

    visit[index] = 1;

    q.push(index);


    while (!q.empty()) {

        int current = q.front();

        q.pop();

        cout << city[current] << " -> ";


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

```

```

        if (a[current][i] != 0 && visit[i] == 0) {

            visit[i] = 1;

            q.push(i);

        }

    }

}

cout << "END\n";

}

```

```

void Graph::DFS() {

    cout << "\n\nDFS Traversal:\n";

    stack<int> s;

    int visit[10] = {0};

    string start;

    int index = -1;

    cout << "Enter starting city: ";

    cin >> start;

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

        if (start == city[i]) {

            index = i;

            break;

        }

    }

    if (index == -1) {

        cout << "City not found.\n";

        return;

    }

    s.push(index);

```

```

while (!s.empty()) {

    int current = s.top();

    s.pop();

    if (visit[current] == 0) {

        cout << city[current] << " -> ";

        visit[current] = 1;

    }

    for (int i = n - 1; i >= 0; i--) {

        if (a[current][i] != 0 && visit[i] == 0) {

            s.push(i);

        }

    }

}

cout << "END\n";

}

int main() {

    Graph g1;

    int choice;

    do {

        cout << "\n\nGRAPH TRAVERSAL MENU";

        cout << "\n1. Input data";

        cout << "\n2. Display adjacency matrix";

        cout << "\n3. DFS Traversal";

        cout << "\n4. BFS Traversal";

        cout << "\n5. Exit";

        cout << "\nEnter your choice: ";

```

```
cin >> choice;
```

```
switch (choice) {
```

```
    case 1: g1.input(); break;
```

```
    case 2: g1.display(); break;
```

```
    case 3: g1.DFS(); break;
```

```
    case 4: g1.BFS(); break;
```

```
    case 5: cout << "Exiting program.\n"; break;
```

```
    default: cout << "Invalid choice. Try again!\n";
```

```
}
```

```
} while (choice != 5);
```

```
return 0;
```

```
}
```

```
main.cpp  Run  Output  Clear
```

```
138 Graph g1;
139 int choice;
140
141 do {
142     cout << "\n\nGRAPH TRAVERSAL MENU";
143     cout << "\n1. Input data";
144     cout << "\n2. Display adjacency matrix";
145     cout << "\n3. DFS Traversal";
146     cout << "\n4. BFS Traversal";
147     cout << "\n5. Exit";
148     cout << "\nEnter your choice: ";
149     cin >> choice;
150
151     switch (choice) {
152         case 1: g1.input(); break;
153         case 2: g1.display(); break;
154         case 3: g1.DFS(); break;
155         case 4: g1.BFS(); break;
156         case 5: cout << "Exiting program.\n"; break;
157         default: cout << "Invalid choice. Try again!\n";
158     }
159 } while (choice != 5);
160
161 return 0;
162 }
163
```

```
GRAPH TRAVERSAL MENU
1. Input data
2. Display adjacency matrix
3. DFS Traversal
4. BFS Traversal
5. Exit
Enter your choice: 1

Enter number of cities: 2

Enter names of cities:
City 1: pune
City 2: pimpri

Enter distances between cities:
Distance from pune to pimpri: 25km
Distance from pimpri to pune:

GRAPH TRAVERSAL MENU
1. Input data
2. Display adjacency matrix
3. DFS Traversal
4. BFS Traversal
5. Exit
```