LAB 15

SP22-BSCS-0046
M. HUZAIFA MUSTAFA
SECTION AM

SOURCE CODE:

```
#include <iostream>

#include <vector>

#include <list>

using namespace std;


const int MAX = 100;


class Graph {

public:

    int adj_matrix[MAX][MAX];

    vector<int> adjLists[MAX];

    bool* visited;

    int n;


    Graph() {

        cout << "Input Vertices: "<<endl;

        cin >> n;

        // Initialize the adjacency matrix with zeros

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

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

```
            adj_matrix[i][j] = 0;
        }
    }
}

// Print the adjacencymatrix
void PrintMatrix() {
    cout << "   ";
    for (int i = 0; i < n; i++) {
        cout << i + 1 << " ";
    }
    cout << endl;
    for (int i = 0; i < n; i++) {
        cout << i + 1 << " : ";
        for (int j = 0; j < n; j++) {
            cout << adj_matrix[i][j] << " ";
        }
        cout << endl;
    }
}

// Add an edge to the graph
void add_edge(int source, int destination, int type) {
    if (adj_matrix[source - 1][destination - 1] == 1) {
        cout << "Edge already exists." << endl;
```

```
    } else {
        adj_matrix[source - 1][destination - 1] = 1;
        if (type == 2) {
            adj_matrix[destination - 1][source - 1] = 1;
        }
        cout << "Edge added successfully." << endl;
    }
}

// Remove an edge from the graph
void remove_edge(int source, int destination, int type) {
    if (adj_matrix[source - 1][destination - 1] == 0) {
        cout << "Edge does not exist." << endl;
    } else {
        adj_matrix[source - 1][destination- 1] = 0;
        if (type == 2) {
            adj_matrix[destination - 1][source - 1] = 0;
        }
        cout << "Edge removed successfully." << endl;
    }
}

// Add an edge to the graph using adjacency list
void addEdge(vector<int> adj[], int s, int d) {
    adj[s].push_back(d);
```

```
    adj[d].push_back(s);
}

// Print the adjacency list
void printGraph(vector<int> adj[], int V) {
    for (int d = 0; d < V; ++d) {
        cout << "\n Vertex " << d << ":";
        for (auto x : adj[d]) {
            cout << "-> " << x;
        }
        cout << endl;
    }
}

void BFS(int startVertex) {
    visited = new bool[n];
    for (int i = 0; i < n; i++)
        visited[i] = false;

    list<int> queue;

    visited[startVertex] = true;
    queue.push_back(startVertex);

    list<int>::iterator i;
```

```
    while (!queue.empty()) {
        int currVertex = queue.front();
        cout <<currVertex<<" , ";
        queue.pop_front();

        for (int j = 0; j < n; j++) {
            if (adj_matrix[currVertex][j] &&!visited[j]) {
                visited[j] = true;
                queue.push_back(j);
            }
        }
    }
}

void DFS(int vertex) {
     visited[vertex] = true;
    cout << vertex+1<< " ";

    for (int j = 0; j < n; j++) {
        if (adj_matrix[vertex][j] && !visited[j]) {
            DFS(j);
        }
    }
}
```

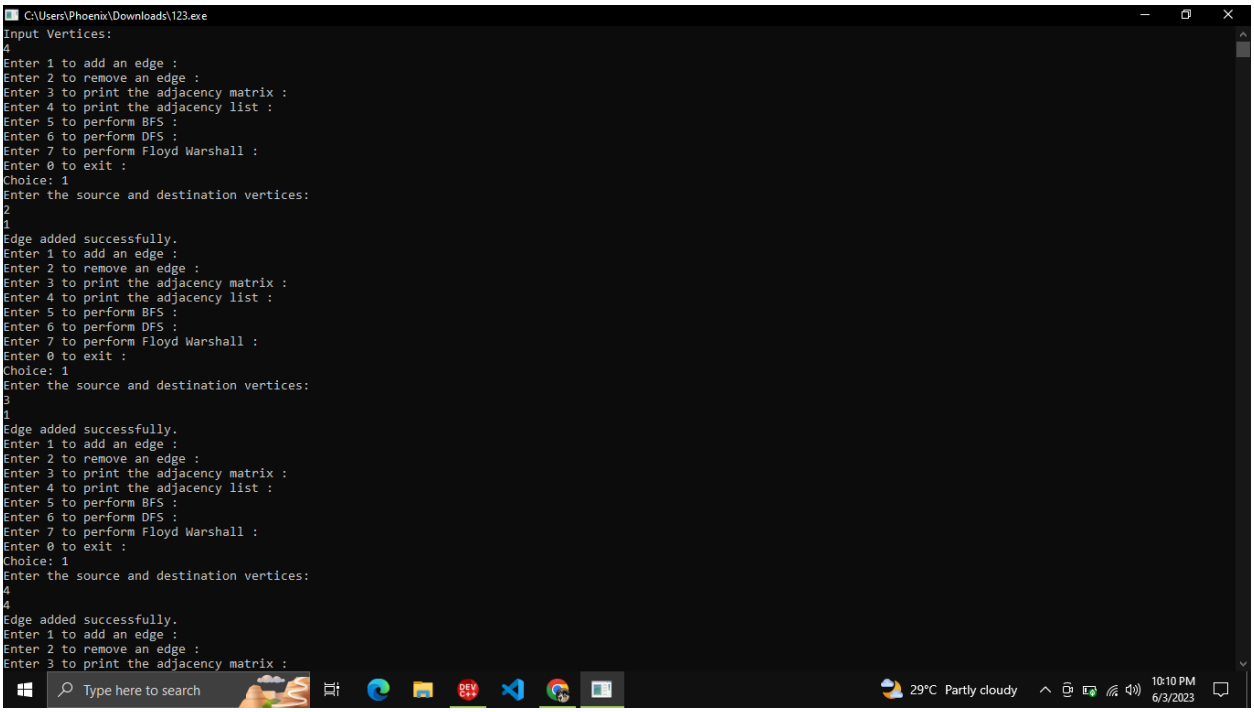
```
void floydWarshall() {
    int dist[n][n];
    int predecessor[n][n];

   // Initialize the distance matrix
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < n; j++) {
         dist[i][j] = adj_matrix[i][j];
         predecessor[i][j] = i;
      }
   }

   // Calculate the shortest path between all pairs of vertices
   for (int k = 0; k < n; k++) {
      for (int i = 0; i < n; i++) {
         for (int j = 0; j < n; j++) {
            if (dist[i][k] + dist[k][j] < dist[i][j]) {
               dist[i][j] = dist[i][k] + dist[k][j];
               predecessor[i][j] = predecessor[k][j];
            }
         }
      }
   }
```

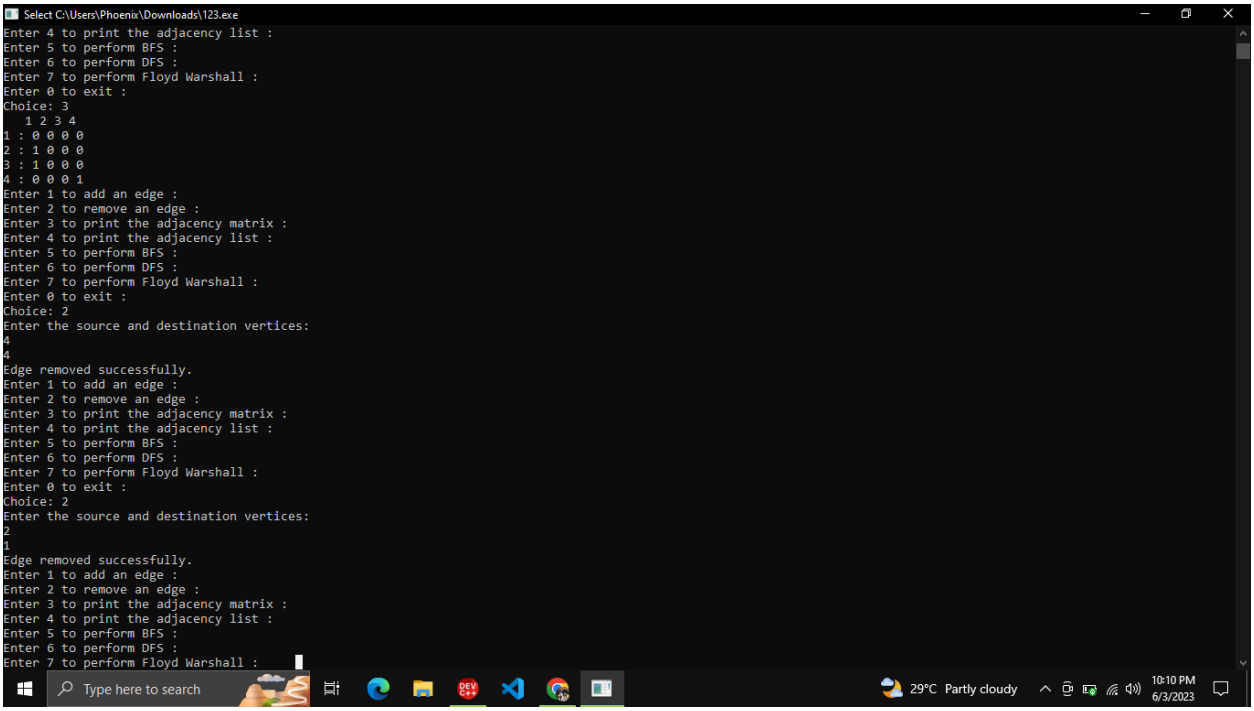
```
    // Print the shortest distances
    cout << "Shortest distances between all pairs of vertices: " << endl;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (dist[i][j] == INT_MAX) {
                cout << "INF ";
            } else {
                cout << dist[i][j] << " ";
            }
        }
        cout << endl;
    }
}



};

int main() {
    Graph g1;
    int choice;
    int typegraph = 1;
    int sourceVertex;
    int destinationVertex;
    vector<int> adj[MAX];
```

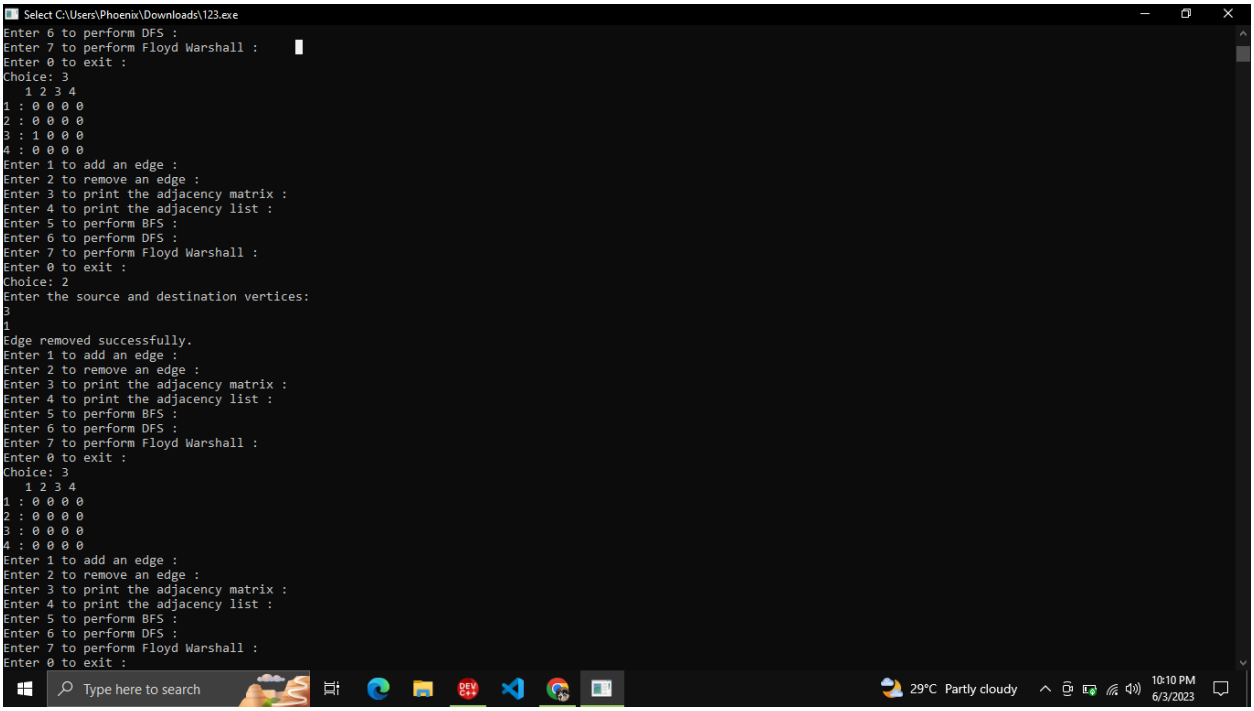
```
do {
    cout << "Enter 1 to add an edge : " << endl;
    cout << "Enter 2 to remove an edge : " << endl;
    cout << "Enter 3 to print the adjacency matrix : " << endl;
    cout << "Enter 4 to print the adjacency list : " << endl;
    cout << "Enter 5 to perform BFS : " << endl;
    cout << "Enter 6 to perform DFS : " << endl;
    cout << "Enter 7 to perform Floyd Warshall : " << endl;
    cout << "Enter 0 to exit : " << endl;
    cout << "Choice: ";
    cin >> choice;

    if (choice == 1) {
        cout << "Enter the source and destination vertices: "<<endl;
        cin >> sourceVertex >> destinationVertex;
        g1.add_edge(sourceVertex, destinationVertex, typegraph);
        g1.addEdge(adj, sourceVertex-1, destinationVertex-1);
    } else if (choice == 2) {
        cout << "Enter the source and destination vertices: "<<endl;
        cin >> sourceVertex >> destinationVertex;
        g1.remove_edge(sourceVertex, destinationVertex, typegraph);
    } else if (choice ==3) {
        g1.PrintMatrix();
    } else if (choice == 4) {
        g1.printGraph(adj, g1.n);
```

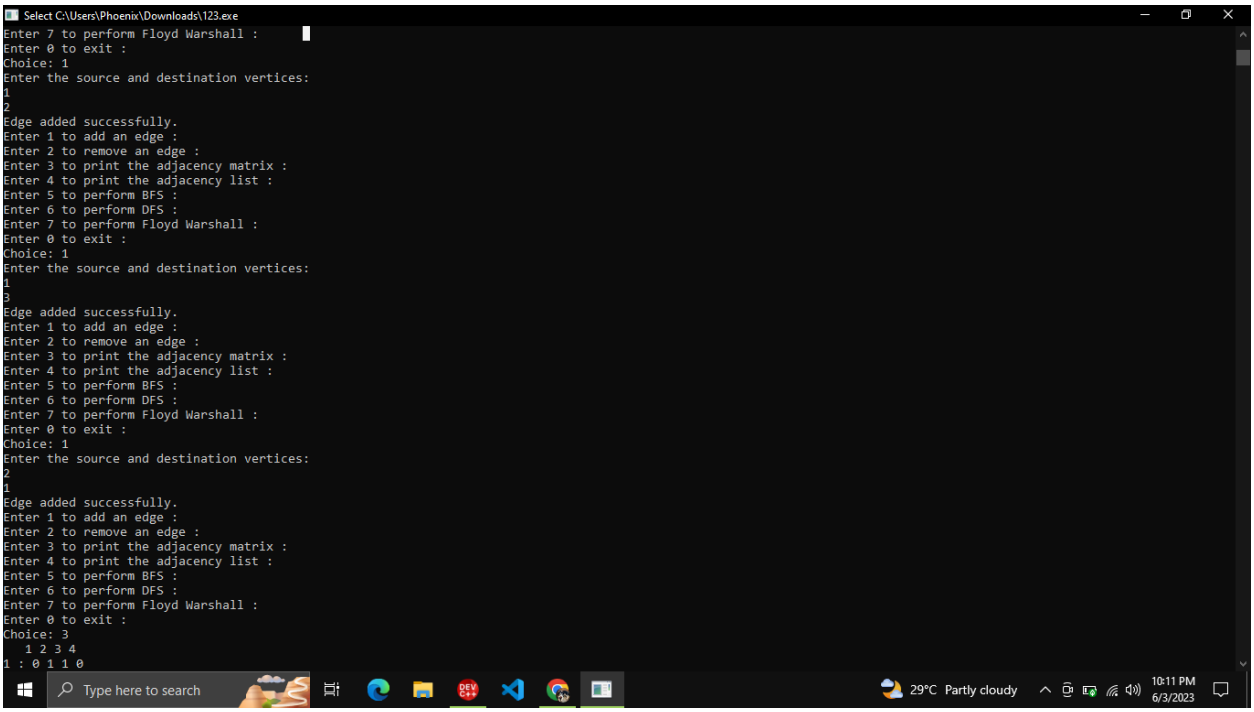
```
        } else if (choice == 5) {
            int startVertex;
            cout << "Enter the starting vertex for BFS: "<<endl;
            cin >> startVertex;
            cout << "BFS traversal: "<<endl;
            g1.BFS(startVertex - 1);
            cout << endl;
        } else if (choice == 6) {
            int startVertex;
            cout << "Enter the starting vertex for DFS: "<<endl;
            cin >> startVertex;
            cout << "DFS traversal: "<<endl;
            g1.visited = new bool[g1.n];
            for (int i = 0; i < g1.n; i++)
                g1.visited[i] = false;
            g1.DFS(startVertex - 1);
            cout << endl;
        } else if(choice == 7){
        g1.floydWarshall();
}

    } while (choice != 0);

    return 0;
}
```

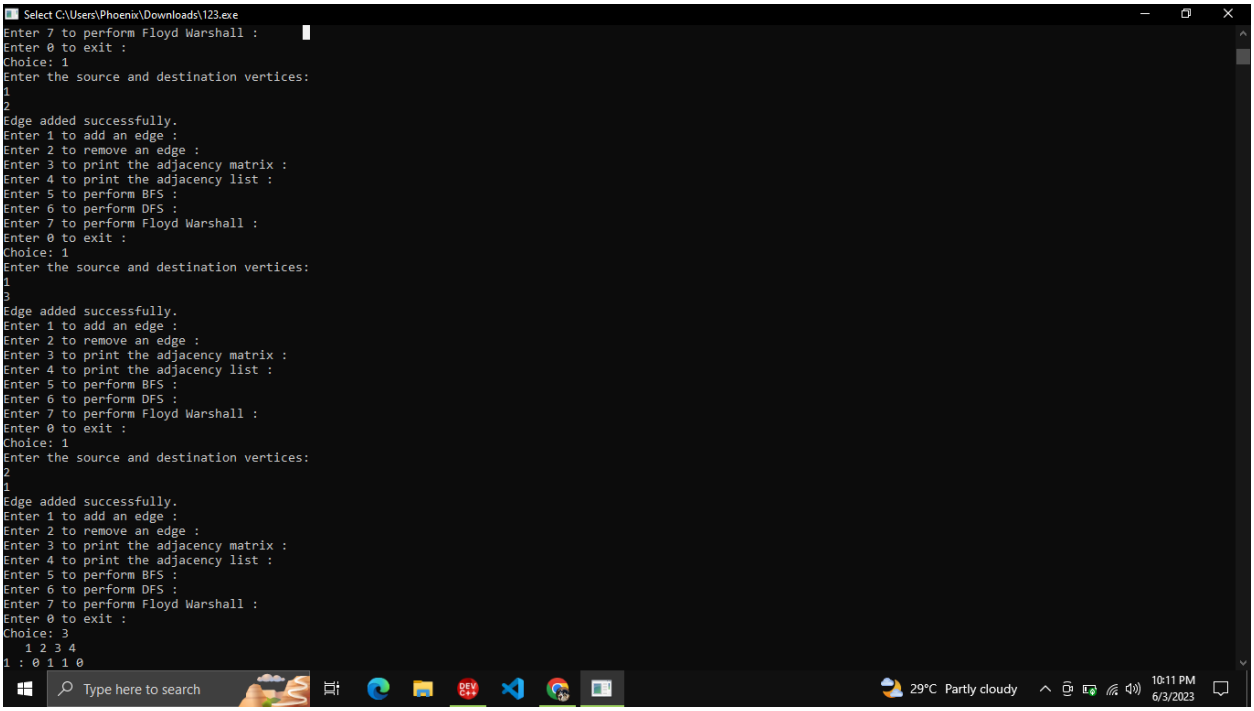
PICTURE:

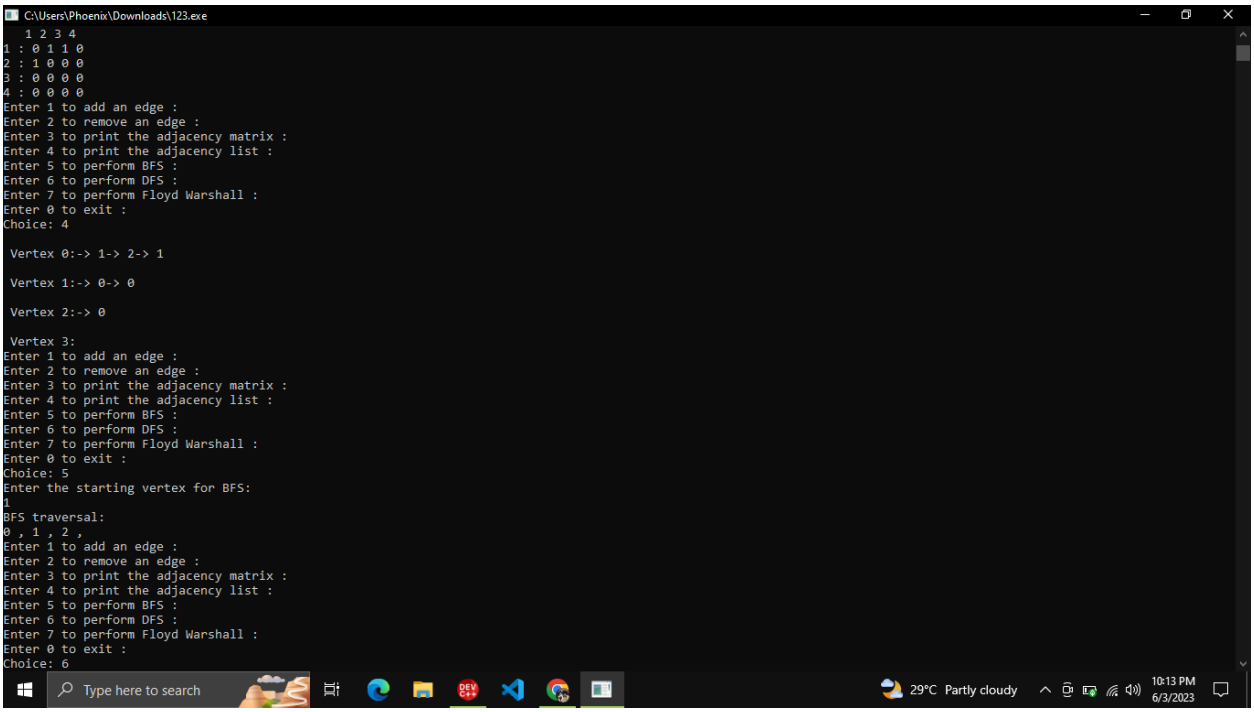

C:\Users\Phoenix\Downloads\123.exe
Input Vertices:
4
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
2
1
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
3
1
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
4
4
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Type here to search
29°C Partly cloudy
10:10 PM
6/3/2023

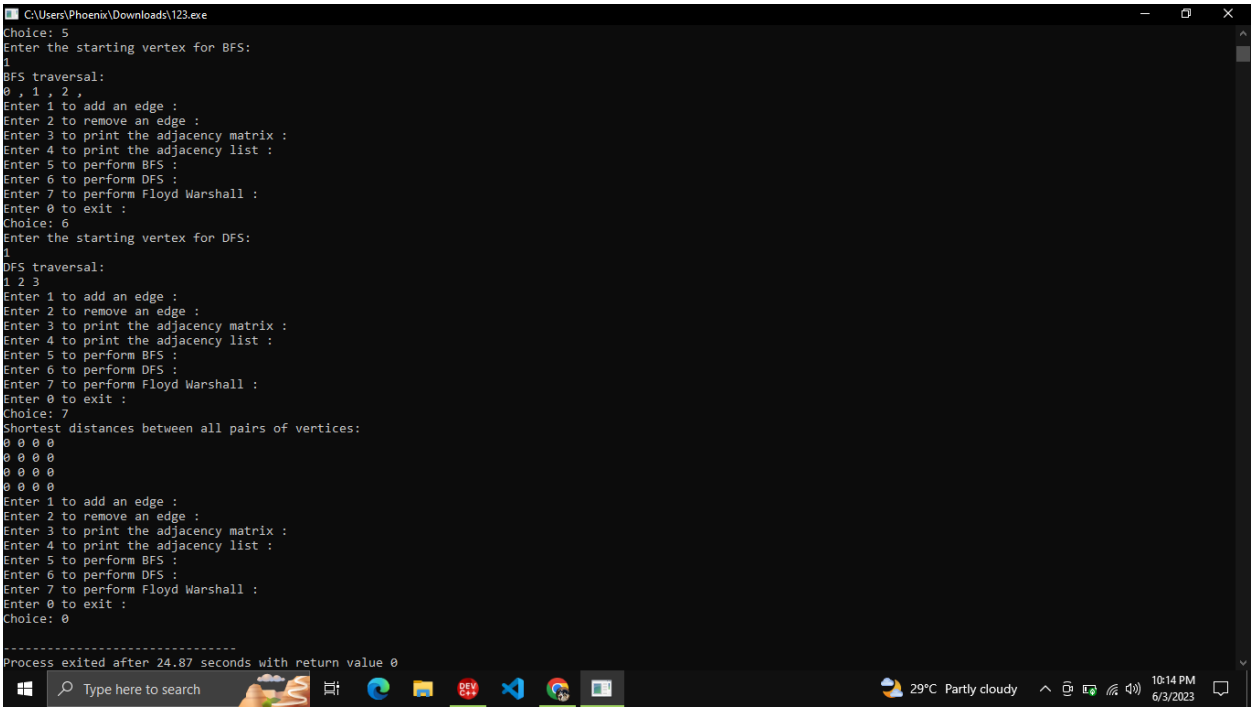

Select C:\Users\Phoenix\Downloads\123.exe
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 3
1 2 3 4
1 : 0 0 0 0
2 : 1 0 0 0
3 : 1 0 0 0
4 : 0 0 0 1
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 2
Enter the source and destination vertices:
4
4
Edge removed successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 2
Enter the source and destination vertices:
2
1
Edge removed successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Type here to search
29°C Partly cloudy
10:10 PM
6/3/2023


Select C:\Users\Phoenix\Downloads\123.exe
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 3
1 2 3 4
1 : 0 0 0 0
2 : 0 0 0 0
3 : 1 0 0 0
4 : 0 0 0 0
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 2
Enter the source and destination vertices:
3
1
Edge removed successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 3
1 2 3 4
1 : 0 0 0 0
2 : 0 0 0 0
3 : 0 0 0 0
4 : 0 0 0 0
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Type here to search
29°C Partly cloudy
10:10 PM
6/3/2023


Select C:\Users\Phoenix\Downloads\123.exe
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
1
2
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
1
3
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
2
1
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 3
1 2 3 4
1 : 0 1 1 0
Type here to search
29°C Partly cloudy
10:11 PM
6/3/2023


Select C:\Users\Phoenix\Downloads\123.exe
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
1
2
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
1
3
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 1
Enter the source and destination vertices:
2
1
Edge added successfully.
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 3
1 2 3 4
1 : 0 1 1 0
Type here to search
29°C Partly cloudy
10:11 PM
6/3/2023


C:\Users\Phoenix\Downloads\123.exe
1 2 3 4
1 : 0 1 1 0
2 : 1 0 0 0
3 : 0 0 0 0
4 : 0 0 0 0
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 4
Vertex 0:-> 1-> 2-> 1
Vertex 1:-> 0-> 0
Vertex 2:-> 0
Vertex 3:
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 5
Enter the starting vertex for BFS:
1
BFS traversal:
0 , 1 , 2 ,
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 6
Type here to search
29°C Partly cloudy
10:13 PM
6/3/2023


C:\Users\Phoenix\Downloads\123.exe
Choice: 5
Enter the starting vertex for BFS:
1
BFS traversal:
0 , 1 , 2 ,
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 6
Enter the starting vertex for DFS:
1
DFS traversal:
1 2 3
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 7
Shortest distances between all pairs of vertices:
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
Enter 1 to add an edge :
Enter 2 to remove an edge :
Enter 3 to print the adjacency matrix :
Enter 4 to print the adjacency list :
Enter 5 to perform BFS :
Enter 6 to perform DFS :
Enter 7 to perform Floyd Warshall :
Enter 0 to exit :
Choice: 0

Process exited after 24.87 seconds with return value 0
Type here to search
29°C Partly cloudy
10:14 PM
6/3/2023