Q16)

#include <iostream>

using namespace std;

class Graph

{

int n;

int \*\*arr;

public:

Graph(int a)

{

n = a;

arr = new int \*[n];

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

arr[i] = new int[n];

}

void inputGraph()

{

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

{

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

cin >> arr[i][j];

}

}

bool isComplete()

{

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

{

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

{

if (i == j && arr[i][j])

return false;

else if (i != j && (!arr[i][j] || !arr[j][i]))

return false;

}

}

return true;

}

};

int main()

{

int size = 3;

Graph G(size);

cout << "Enter the graph matrix" << endl;

G.inputGraph();

if (G.isComplete())

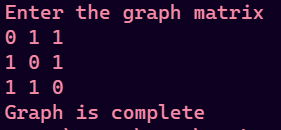
cout << "Graph is complete" << endl;

else

cout << "Graph is complete" << endl;

return 0;

}



Q17)

#include <iostream>

using namespace std;

class Graph

{

int n;

int \*\*arr;

public:

Graph(int a)

{

n = a;

arr = new int \*[n];

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

arr[i] = new int[n];

}

void inputGraph()

{

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

{

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

cin >> arr[i][j];

}

}

int indegree(int v)

{ // col v sum

if (v >= n)

return -1; // invalid vertex

int res = 0;

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

res += arr[i][v];

return res;

}

int outdegree(int v)

{ // row v sum

if (v >= n)

return -1; // invalid vertex

int res = 0;

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

res += arr[v][i];

return res;

}

};

int main()

{

int size = 3;

Graph G(size);

cout << "Enter graph matrix" << endl;

G.inputGraph();

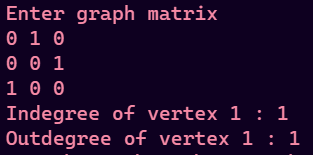
int vertex = 1;

cout << "Indegree of vertex " << vertex << " : " << G.indegree(vertex) << endl;

cout << "Outdegree of vertex " << vertex << " : " << G.outdegree(vertex) << endl;

return 0;

}



Q18)

#include <iostream>

#include <vector>

#include <list>

using namespace std;

class Graph

{

int n;

int \*\*arr;

public:

Graph(int a)

{

n = a;

arr = new int \*[n];

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

arr[i] = new int[n];

}

void inputGraph()

{

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

{

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

cin >> arr[i][j];

}

}

int pathsOfLength(int u, int v, int r)

{

if (u >= n || v >= n)

return 0;

int res[n][n];

int temp[n][n];

// res = arr

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

{

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

res[i][j] = arr[i][j];

}

int s;

for (int i = 0; i < r - 1; i++)

{

// temp = res \* arr

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

{

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

{

s = 0;

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

s += (res[j][l] \* arr[l][k]);

temp[j][k] = s;

}

}

// res = temp

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

{

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

res[i][j] = temp[i][j];

}

}

return res[u][v];

}

};

int main()

{

Graph G(5);

cout << "Enter the graph" << endl;

G.inputGraph();

int u, v, r;

cout << "Enter initial , final nodes and path length" << endl;

cin >> u >> v >> r;

cout << "Required No. of paths: " << G.pathsOfLength(u, v, r) << endl;

return 0;

}

/\*

0 1 1 1 0

1 0 0 1 1

1 0 0 0 1

1 1 0 0 1

0 1 1 1 0

output = 3 for u = 0 , v = 4 , r = 2

\*/

