**HAMILTONIAN**

#include<iostream>

#define NODE 5

using namespace std;

int graph[NODE][NODE] = {

{0, 1, 0, 1, 0},

{1, 0, 1, 1, 1},

{0, 1, 0, 0, 1},

{1, 1, 0, 0, 1},

{0, 1, 1, 1, 0},

};

/\* int graph[NODE][NODE] = {

{0, 1, 0, 1, 0},

{1, 0, 1, 1, 1},

{0, 1, 0, 0, 1},

{1, 1, 0, 0, 0},

{0, 1, 1, 0, 0},

}; \*/

int path[NODE];

void displayCycle() {

cout<<"Cycle: ";

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

cout << path[i] << " ";

cout << path[0] << endl; //print the first vertex again

}

bool isValid(int v, int k) {

if (graph [path[k-1]][v] == 0) //if there is no edge

return false;

for (int i = 0; i < k; i++) //if vertex is already taken, skip that

if (path[i] == v)

return false;

return true;

}

bool cycleFound(int k) {

if (k == NODE) { //when all vertices are in the path

if (graph[path[k-1]][ path[0] ] == 1 )

return true;

else

return false;

}

for (int v = 1; v < NODE; v++) { //for all vertices except starting point

if (isValid(v,k)) { //if possible to add v in the path

path[k] = v;

if (cycleFound (k+1) == true)

return true;

path[k] = -1; //when k vertex will not in the solution

}

}

return false;

}

bool hamiltonianCycle() {

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

path[i] = -1;

path[0] = 0; //first vertex as 0

if ( cycleFound(1) == false ) {

cout << "Solution does not exist"<<endl;

return false;

}

displayCycle();

return true;

}

int main()

{

hamiltonianCycle();

}

