#include<stdio.h>

#define V 5

void printSolution(int path[]);

bool isSafe(int v, bool graph[V][V], int path[], int pos)

{

if (graph [ path[pos-1] ][ v ] == 0)

return false;

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

if (path[i] == v)

return false;

return true;

}

bool hamCycleUtil(bool graph[V][V], int path[], int pos)

{

if (pos == V)

{

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

return true;

else

return false;

}

for (int v = 1; v < V; v++)

{

if (isSafe(v, graph, path, pos))

{

path[pos] = v;

/\* recur to construct rest of the path \*/

if (hamCycleUtil (graph, path, pos+1) == true)

return true;

/\* If adding vertex v doesn't lead to a solution,

then remove it \*/

path[pos] = -1;

}

}

/\* If no vertex can be added to Hamiltonian Cycle constructed so far,

then return false \*/

return false;

}

bool hamCycle(bool graph[V][V])

{

int \*path = new int[V];

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

path[i] = -1;

path[0] = 0;

if ( hamCycleUtil(graph, path, 1) == false )

{

printf("\nSolution does not exist");

return false;

}

printSolution(path);

return true;

}

void printSolution(int path[])

{

printf ("Solution Exists:"

" Following is one Hamiltonian Cycle \n");

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

printf(" %d ", path[i]);

printf(" %d ", path[0]);

printf("\n");

}

int main()

{

bool graph1[V][V] = {{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},

};

hamCycle(graph1);

bool graph2[V][V] = {{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},

};

hamCycle(graph2);

return 0;

}

