Practical-7

Name: Ujjwal Tiwari

Section :A2

Batch :B4

Roll : 51

Aim: Implement Hamiltonian Cycle using Backtracking.

Code:

#include <stdio.h>

#include <stdbool.h>

#define V 5

void printSolution(int path[]) {

    printf("Hamiltonian Cycle Found:\n");

    for (int i = 0; i < V; i++)

        printf("%c -> ", 'A' + path[i]);

    printf("%c\n", 'A' + path[0]);

}

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;

        return false;

    }

    for (int v = 1; v < V; v++) {

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

            path[pos] = v;

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

                return true;

            path[pos] = -1;

        }

    }

    return false;

}

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

    int path[V];

    for (int i = 0; i < V; i++)

        path[i] = -1;

    path[0] = 0;

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

        printf("No Hamiltonian Cycle exists.\n");

        return false;

    }

    printSolution(path);

    return true;

}

int main() {

    bool graph[V][V] = {

        {0, 1, 1, 0, 1},

        {1, 0, 1, 1, 0},

        {1, 1, 0, 1, 0},

        {0, 1, 1, 0, 1},

        {1, 0, 0, 1, 0}

    };

    hamCycle(graph);

    return 0;

}

Output:-

