DAA Practical 7

Name- Anupama Sanjeevan
Section- A6/B3
Rollno- 49

Aim: Implement Hamiltonian Cycle using Backtracking.
Problem Statement:
The Smart City Transportation Department is designing a night-patrol route for security vehicles.
Each area of the city is represented as a vertex in a graph, and a road between two areas is represented as an edge.
The goal is to find a route that starts from the main headquarters (Area A), visits each area exactly once, and returns back to the headquarters — forming a Hamiltonian Cycle.
If such a route is not possible, display a suitable message.
1) Adjacency Matrix
A B C D E
A 0 1 1 0 1
B 1 0 1 1 0
C 1 1 0 1 0
D 0 1 1 0 1
E 1 0 0 1 0

1) Adjacency Matrix
T M S H C
T 0 1 1 0 1
M 1 0 1 1 0
S 1 1 0 1 1
H 0 1 1 0 1
C 1 0 1 1 0

CODE-
```
#include <stdio.h>

#define V 5

int isSafe(int v, int graph[V][V], int path[], int pos) {
	if (graph[path[pos - 1]][v] == 0)
	return 0;
	for (int i = 0; i < pos; i++)
	if (path[i] == v)
```

```
        return 0;
        return 1;
}
int solveHamiltonian(int graph[V][V], int path[], int pos) {
        if (pos == V) {
        if (graph[path[pos - 1]][path[0]] == 1)
        return 1;
        return 0;
        }
        for (int v = 1; v < V; v++) {
        if (isSafe(v, graph, path, pos)) {
        path[pos] = v;
        if (solveHamiltonian(graph, path, pos + 1) == 1)
                return 1;
        path[pos] = -1;
        }
        }
        return 0;
}
void printSolution(int path[], char names[]) {
   printf("Hamiltonian Cycle found:\n");
        for (int i = 0; i < V; i++)
      printf("%c -> ", names[path[i]]);
   printf("%c\n", names[path[0]]);
}
void hamiltonianCycle(int graph[V][V], char names[]) {
        int path[V];
        for (int i = 0; i < V; i++)
        path[i] = -1;
        path[0] = 0;

        if (solveHamiltonian(graph, path, 1) == 0)
      printf("No Hamiltonian Cycle exists\n");
        else
      printSolution(path, names);
}

int main() {
        int graph1[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}
```

```
    };
    char names1[] = {'A', 'B', 'C', 'D', 'E'};

    int graph2[V][V] = {
    {0, 1, 1, 0, 1},
    {1, 0, 1, 1, 0},
    {1, 1, 0, 1, 1},
    {0, 1, 1, 0, 1},
    {1, 0, 1, 1, 0}
    };
    char names2[] = {'T', 'M', 'S', 'H', 'C'};

    printf("Graph 1 (Areas A–E):\n");
  hamiltonianCycle(graph1, names1);

  printf("\nGraph 2 (Areas T–C):\n");
  hamiltonianCycle(graph2, names2);

    return 0;
}
```

OUTPUT-

```
Graph 1 (Areas A to E):
Hamiltonian Cycle found:
A -> B -> C -> D -> E -> A

Graph 2 (Areas T to C):
Hamiltonian Cycle found:
T -> M -> S -> H -> C -> T
```

GFG: