Practical 7

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

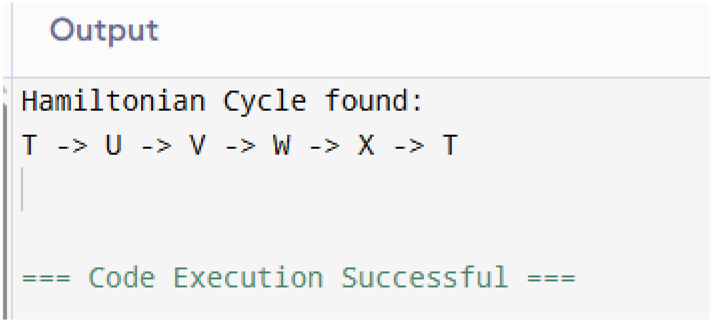
C 1 0 1 1 0

Code:

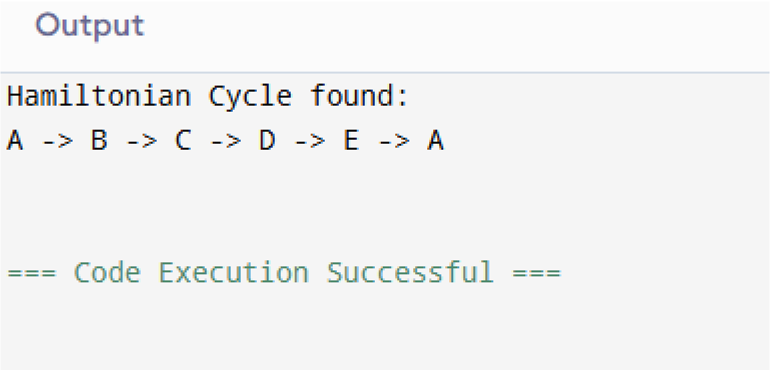
```
#include <stdio.h>
#include <stdbool.h>

#define MAX 100

int V = 5;



int graph[MAX][MAX] = {
    {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}
};

int path[MAX];

bool isSafe(int v, 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 hamiltonianCycle(int pos) {
    if (pos == V) {
        return graph[path[pos - 1]][path[0]] == 1;
    }

    for (int v = 1; v < V; v++) {
        if (isSafe(v, pos)) {
            path[pos] = v;
            if (hamiltonianCycle(pos + 1))
                return true;
            path[pos] = -1;
        }
    }

    return false;
}

int main() {
    for (int i = 0; i < V; i++)
        path[i] = -1;

    path[0] = 0;

    if (hamiltonianCycle(1)) {
        printf("Hamiltonian Cycle found:\n");
        for (int i = 0; i < V; i++) {
            char label = 'T' + path[i];
            printf("%c -> ", label);
        }
        printf("T\n");

    } else {
        printf("No Hamiltonian Cycle exists for the given city layout.\n");
    }

    return 0;
}
```

OUTPUT:



```
Hamiltonian Cycle found:
T -> U -> V -> W -> X -> T


=== Code Execution Successful ===
```



```
Hamiltonian Cycle found:
A -> B -> C -> D -> E -> A


=== Code Execution Successful ===
```