Design and Analysis of Algorithms Lab

Name: Arya Bodkhe

Section-A4 Batch: B-1

Roll No.:09

PRACTICAL NO. 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 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

2) Adjacency Matrix

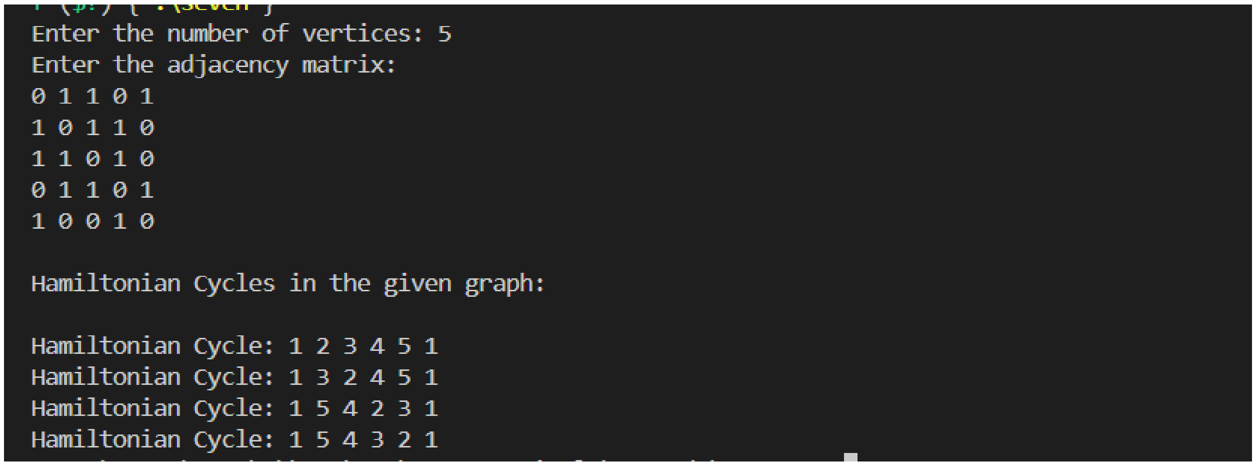
	T	M	S	H	C
T	0	1	1	0	1
M	1	0	1	1	0
S	1	1	0	1	1
H	0	1	1	0	1
C	1	0	1	1	0

CODE:

```c
#include<stdio.h>
#define MAX 10

void NextValue(int k, int n, int G[MAX][MAX], int x[MAX]) {
    int j;
    while (1) {
        x[k] = (x[k] + 1) % (n + 1);
        if (x[k] == 0)
            return;

        if (G[x[k - 1]][x[k]] != 0) {
            for (j = 1; j < k; j++)
                if (x[j] == x[k])
                    break;

            if (j == k) {
                if (k < n || (k == n && G[x[n]][x[1]] != 0))
                    return;
            }
        }
    }
}

void Hamiltonian(int k, int n, int G[MAX][MAX], int x[MAX]) {
    while (1) {
        NextValue(k, n, G, x);
        if (x[k] == 0)
            return;

        if (k == n) {
            printf("\nHamiltonian Cycle: ");
            for (int i = 1; i <= n; i++)
                printf("%d ", x[i]);
            printf("%d", x[1]);
        } else {
            Hamiltonian(k + 1, n, G, x);
        }
    }
}

int main() {
    int n;
    int G[MAX][MAX];
    int x[MAX];
    int i, j;

    printf("Enter the number of vertices: ");
    scanf("%d", &n);

    printf("Enter the adjacency matrix:\n");
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            scanf("%d", &G[i][j]);

    for (i = 1; i <= n; i++)
        x[i] = 0;

    x[1] = 1;

    printf("\nHamiltonian Cycles in the given graph:\n");
    Hamiltonian(2, n, G, x);

    return 0;
}
```

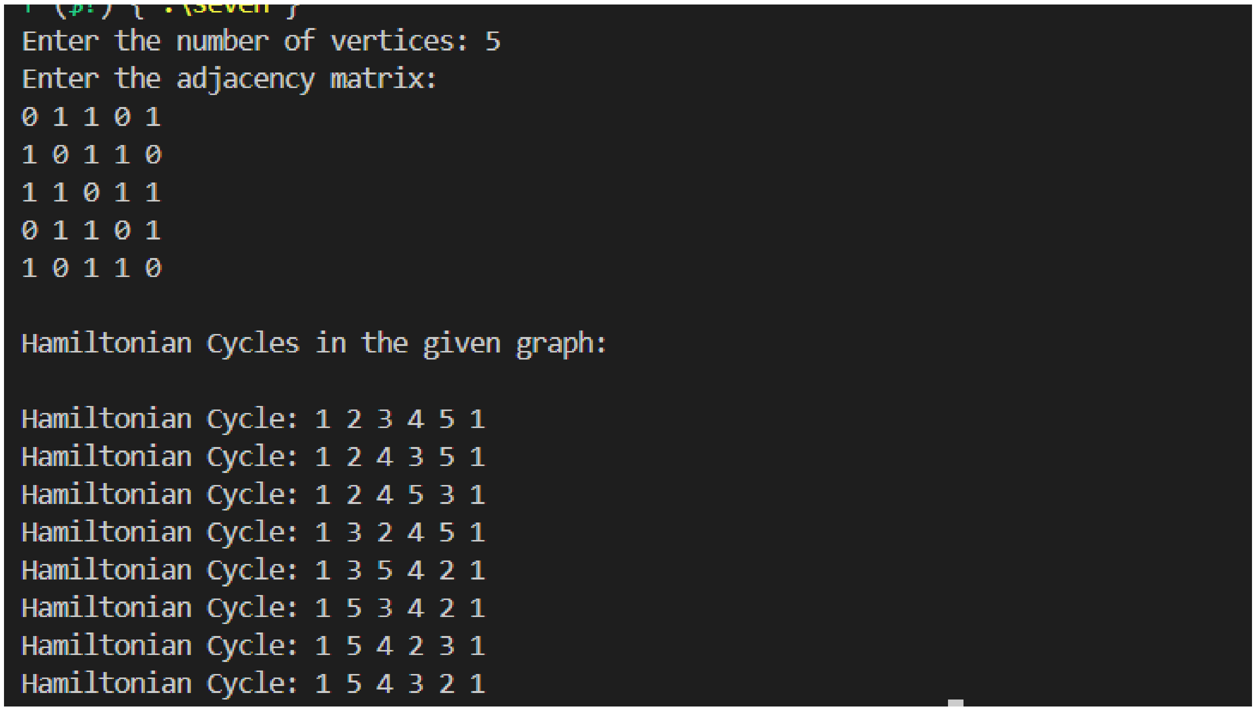
OUTPUT:

1)



```
Enter the number of vertices: 5
Enter the adjacency matrix:
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

Hamiltonian Cycles in the given graph:

Hamiltonian Cycle: 1 2 3 4 5 1
Hamiltonian Cycle: 1 3 2 4 5 1
Hamiltonian Cycle: 1 5 4 2 3 1
Hamiltonian Cycle: 1 5 4 3 2 1
```

2)



```
Enter the number of vertices: 5
Enter the adjacency matrix:
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

Hamiltonian Cycles in the given graph:

Hamiltonian Cycle: 1 2 3 4 5 1
Hamiltonian Cycle: 1 2 4 3 5 1
Hamiltonian Cycle: 1 2 4 5 3 1
Hamiltonian Cycle: 1 3 2 4 5 1
Hamiltonian Cycle: 1 3 5 4 2 1
Hamiltonian Cycle: 1 5 3 4 2 1
Hamiltonian Cycle: 1 5 4 2 3 1
Hamiltonian Cycle: 1 5 4 3 2 1
```