Exp_16\HamiltonianCycles.c

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

#define MAX 20

int graph[MAX][MAX];
int path[MAX];
int n;
int cycleCount = 0;

void printPath() {
    printf("Hamiltonian Cycle %d: ", ++cycleCount);
    for (int i = 0; i < n; i++) {
        printf("%d ", path[i]);
    }
    printf("%d\n", path[0]); // Return to starting vertex
}

bool isSafe(int v, int pos) {
    // Check if vertex v is adjacent to the previous vertex
    if (graph[path[pos - 1]][v] == 0)
        return false;

    // Check if vertex v is already in path
    for (int i = 0; i < pos; i++) {
        if (path[i] == v)
            return false;
    }

    return true;
}

bool hamiltonianCycleUtil(int pos) {
    // Base case: if all vertices are included in path
    if (pos == n) {
        // Check if there is an edge from the last vertex to the first vertex
        if (graph[path[pos - 1]][path[0]] == 1) {
            printPath();
            return true; // Continue to find more cycles
        }
        return false;
    }

    // Try different vertices as the next candidate
    for (int v = 1; v < n; v++) {
        if (isSafe(v, pos)) {
            path[pos] = v;

            // Recur to construct rest of the path
            hamiltonianCycleUtil(pos + 1);

```

```
            // Backtrack
            path[pos] = -1;
        }
    }

    return false;
}

void findHamiltonianCycles() {
    // Initialize path array
    for (int i = 0; i < n; i++)
        path[i] = -1;

    // Start from vertex 0
    path[0] = 0;
    cycleCount = 0;

    printf("\nFinding all Hamiltonian Cycles:\n");
    hamiltonianCycleUtil(1);

    if (cycleCount == 0) {
        printf("No Hamiltonian Cycle exists in the given graph.\n");
    } else {
        printf("\nTotal number of Hamiltonian Cycles: %d\n", cycleCount);
    }
}

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

    printf("\nEnter the adjacency matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    printf("\nAdjacency Matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", graph[i][j]);
        }
        printf("\n");
    }

    findHamiltonianCycles();

    return 0;
}
```