Practical Number: 7

Name: Laghima Kejariwal
Section: A3
Batch: B2
Roll Number: 19
Date: 27-10-2025

Code:

First:

```
#include <stdio.h>

#define n 5

int G[n][n] = {
    {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}
};

int x[n];

int NextValue(int k) {
    int j;
    do {
        x[k] = (x[k] + 1) % n;
        if (x[k] == 0) return 0;
        if (G[x[k - 1]][x[k]] != 0) {
            for (j = 0; j < k; j++)
                if (x[j] == x[k])
                    break;
            if (j == k) {
                if ((k < n - 1) || ((k == n - 1) && G[x[k]][x[0]] != 0))
                    return 1;
            }
        }
    } while (1);
}

void Hamiltonian(int k) {
    int flag;
    do {
        flag = NextValue(k);
        if (!flag) return;
        if (k == n - 1) {
            printf("\nHamiltonian Cycle Found: ");
            for (int i = 0; i < n; i++)
                printf("%c -> ", x[i] + 'A');
```

```
            printf("%c\n", x[0] + 'A');
        } else
            Hamiltonian(k + 1);
    } while (1);
}

int main() {
    printf("\nAdjacency Matrix Representation:\n");
    printf("   A B C D E\n");
    printf("A: 0 1 1 0 1\n");
    printf("B: 1 0 1 1 0\n");
    printf("C: 1 1 0 1 0\n");
    printf("D: 0 1 1 0 1\n");
    printf("E: 1 0 0 1 0\n");

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

    Hamiltonian(1);
    return 0;
}
```

Second:

```
#include <stdio.h>

#define n 5

int G[n][n] = {
    {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 x[n];

// NextValue(k)
int NextValue(int k) {
    int j;
    do {
        x[k] = (x[k] + 1) % n; // next vertex
        if (x[k] == 0) return 0; // no vertex left

        // Check edge from previous vertex
        if (G[x[k - 1]][x[k]] != 0) {
            // Check distinctness
            for (j = 0; j < k; j++)
                if (x[j] == x[k])
                    break;

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

// Hamiltonian(k)
void Hamiltonian(int k) {
    int flag;
    do {
        flag = NextValue(k);
        if (!flag) return;
```

```
        if (k == n - 1) {
            printf("\nHamiltonian Cycle Found: ");
            for (int i = 0; i < n; i++)
                printf("%c -> ", x[i] + 'A');
            printf("%c\n", x[0] + 'A');
        } else
            Hamiltonian(k + 1);

    } while (1);
}

int main() {
    printf("\nAdjacency Matrix Representation:\n");
    printf("   T M S H C\n");
    printf("T: 0 1 1 0 1\n");
    printf("M: 1 0 1 1 0\n");
    printf("S: 1 1 0 1 1\n");
    printf("H: 0 1 1 0 1\n");
    printf("C: 1 0 1 1 0\n");

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

    Hamiltonian(1);
    return 0;
}
```

OutPut:

1)

```
Adjacency Matrix Representation:
   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

Hamiltonian Cycle Found: A -> B -> C -> D -> E -> A

Hamiltonian Cycle Found: A -> C -> B -> D -> E -> A

Hamiltonian Cycle Found: A -> E -> D -> B -> C -> A

Hamiltonian Cycle Found: A -> E -> D -> C -> B -> A


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

2)

```
Adjacency Matrix Representation:
   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

Hamiltonian Cycle Found: A -> B -> C -> D -> E -> A

Hamiltonian Cycle Found: A -> B -> D -> C -> E -> A

Hamiltonian Cycle Found: A -> B -> D -> E -> C -> A

Hamiltonian Cycle Found: A -> C -> B -> D -> E -> A

Hamiltonian Cycle Found: A -> C -> E -> D -> B -> A

Hamiltonian Cycle Found: A -> E -> C -> D -> B -> A

Hamiltonian Cycle Found: A -> E -> D -> B -> C -> A

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