Name: Sriram Gopalkrishnan Iyer
Section: A3
Batch: B3
Roll No: 38

Ramdeobaba University, Nagpur
Department of Computer Science and Engineering
Session: 2024-2025

Design and Analysis of Algorithms Lab **III Semester**

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 back to the headquarters — forming a Hamiltonian Cycle.

If such a route is not possible, display a suitable message.

Code:

```
#include <stdio.h>

int n;
int G[10][10];
int x[10];
int found = 0;

void NextVertex(int k) {
    while(1) {
        x[k] = (x[k] + 1) % (n + 1);
        if(x[k] == 0)
            return;

        if (G[x[k - 1]][x[k]] != 0) {
            int j;
            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) {
    while(1) {
        NextVertex(k);
```

```c
        if(x[k] == 0)
            return;

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

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

   printf("Enter the Adjacency Matrix: ");
   for(int i = 1; i <= n; i++) {
       for(int j = 1; j <= n; j++) {
           scanf("%d", &G[i][j]);
       }
   }

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

   x[1] = 1;
   Hamiltonian(2);
```

```c
    if(!found)
        printf("\nNo Hamiltonian Cycle exists.\n");

    return 0;
}
```

Output:

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

```
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 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%
```

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

```
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 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%
```

GFG Output: