DESIGN AND ANALYSIS OF ALGORITHMS
LAB III SEMESTER

Name : Vardhan Ingole

Section : A4_B3

Roll No : 43

Practical No. : 07

Aim : Implement Hamiltonian Cycle using Backtracking.

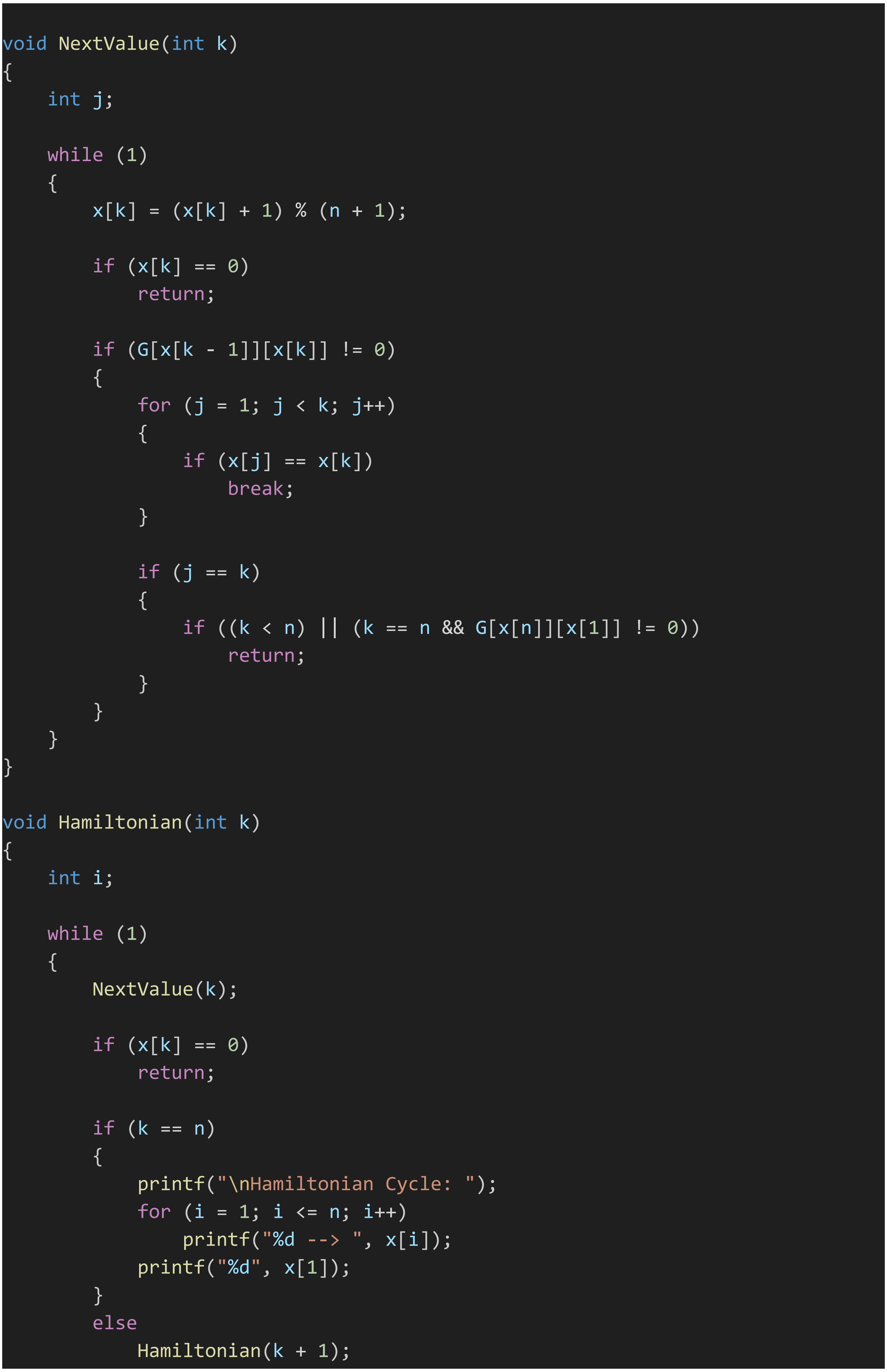
Task 1 :

The Smart City Transportation Department is designing a night-patrol route for security vehicles.

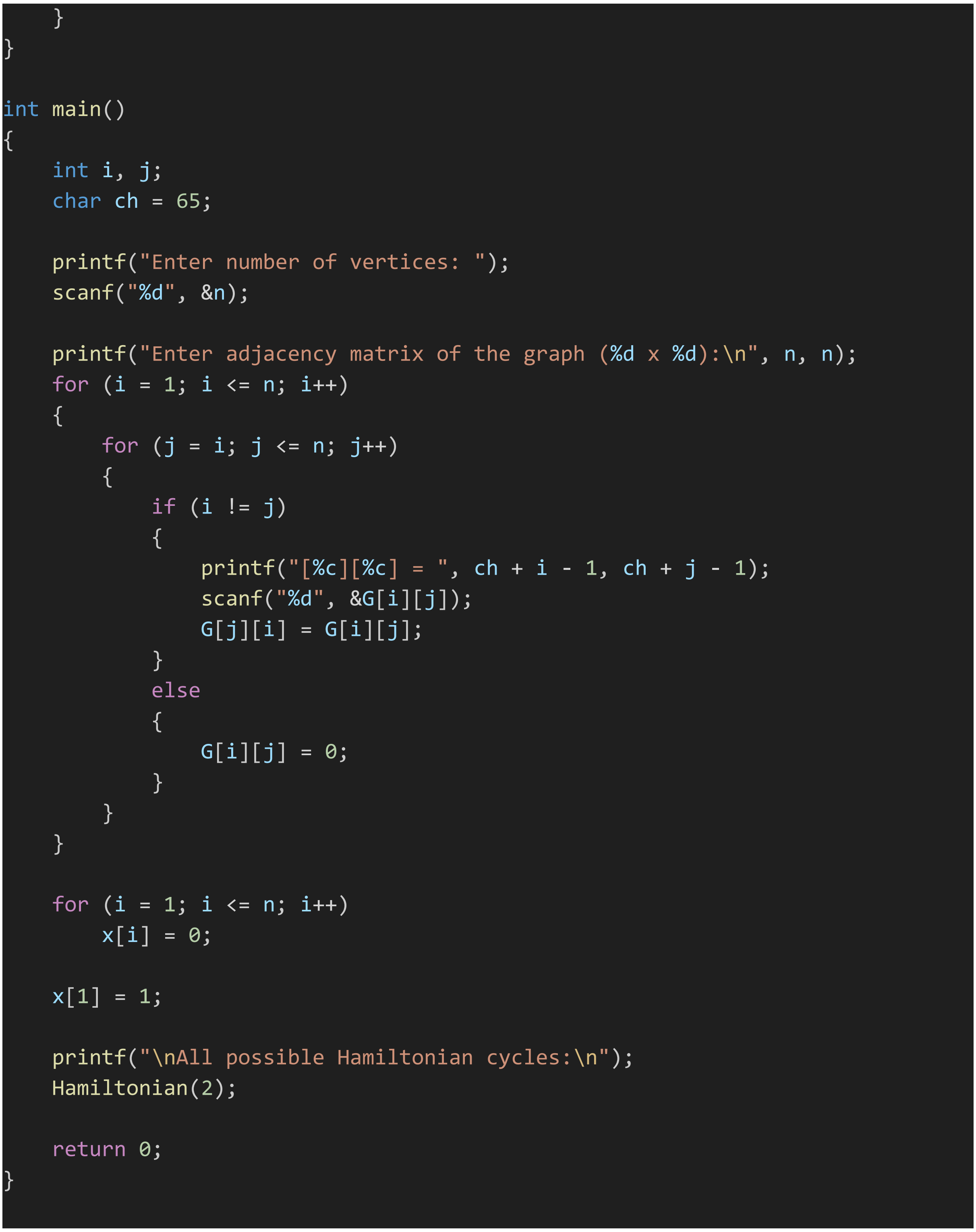
Each area of the city is represented as a vertex in a graph, and a road between two areas is represented as an edge.

Code :

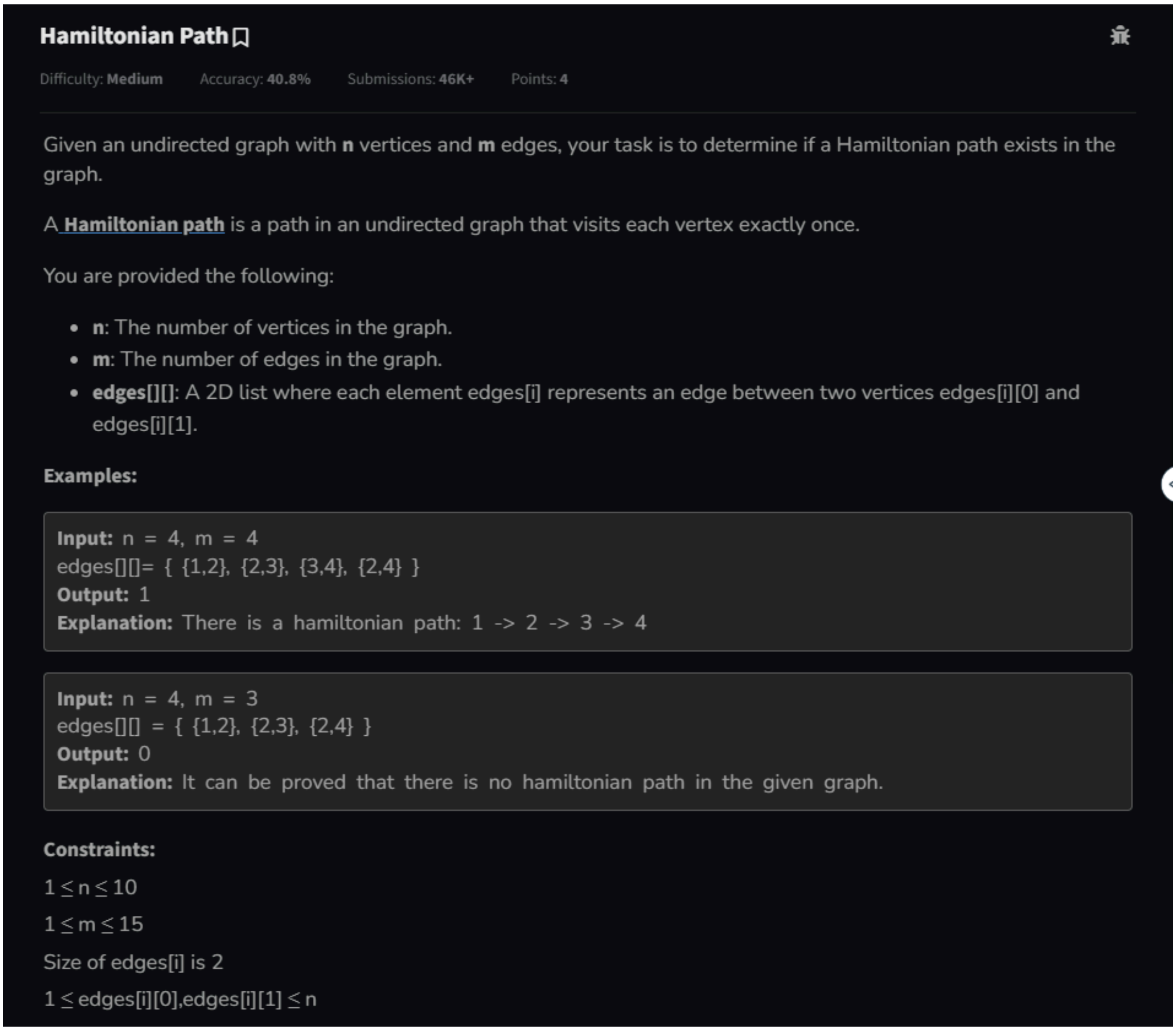
```
#include <stdio.h>

#define MAX 10

int n;
int G[MAX][MAX];
int x[MAX];
```



```
void NextValue(int k)
{
    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 i;

    while (1)
    {
        NextValue(k);

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

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



```c
    }
}

int main()
{
    int i, j;
    char ch = 65;

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

    printf("Enter adjacency matrix of the graph (%d x %d):\n", n, n);
    for (i = 1; i <= n; i++)
    {
        for (j = i; j <= n; j++)
        {
            if (i != j)
            {
                printf("[%c][%c] = ", ch + i - 1, ch + j - 1);
                scanf("%d", &G[i][j]);
                G[j][i] = G[i][j];
            }
            else
            {
                G[i][j] = 0;
            }
        }
    }

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

    x[1] = 1;

    printf("\nAll possible Hamiltonian cycles:\n");
    Hamiltonian(2);

    return 0;
}
```



Hamiltonian Path

Difficulty: **Medium** Accuracy: **40.8%** Submissions: **46K+** Points: **4**

Given an undirected graph with **n** vertices and **m** edges, your task is to determine if a Hamiltonian path exists in the graph.

A **Hamiltonian path** is a path in an undirected graph that visits each vertex exactly once.

You are provided the following:

- **n**: The number of vertices in the graph.
- **m**: The number of edges in the graph.
- **edges[][]**: A 2D list where each element edges[i] represents an edge between two vertices edges[i][0] and edges[i][1].

Examples:

```
Input: n = 4, m = 4
edges[][]= { {1,2}, {2,3}, {3,4}, {2,4} }
Output: 1
Explanation: There is a hamiltonian path: 1 -> 2 -> 3 -> 4
```

```
Input: n = 4, m = 3
edges[][] = { {1,2}, {2,3}, {2,4} }
Output: 0
Explanation: It can be proved that there is no hamiltonian path in the given graph.
```

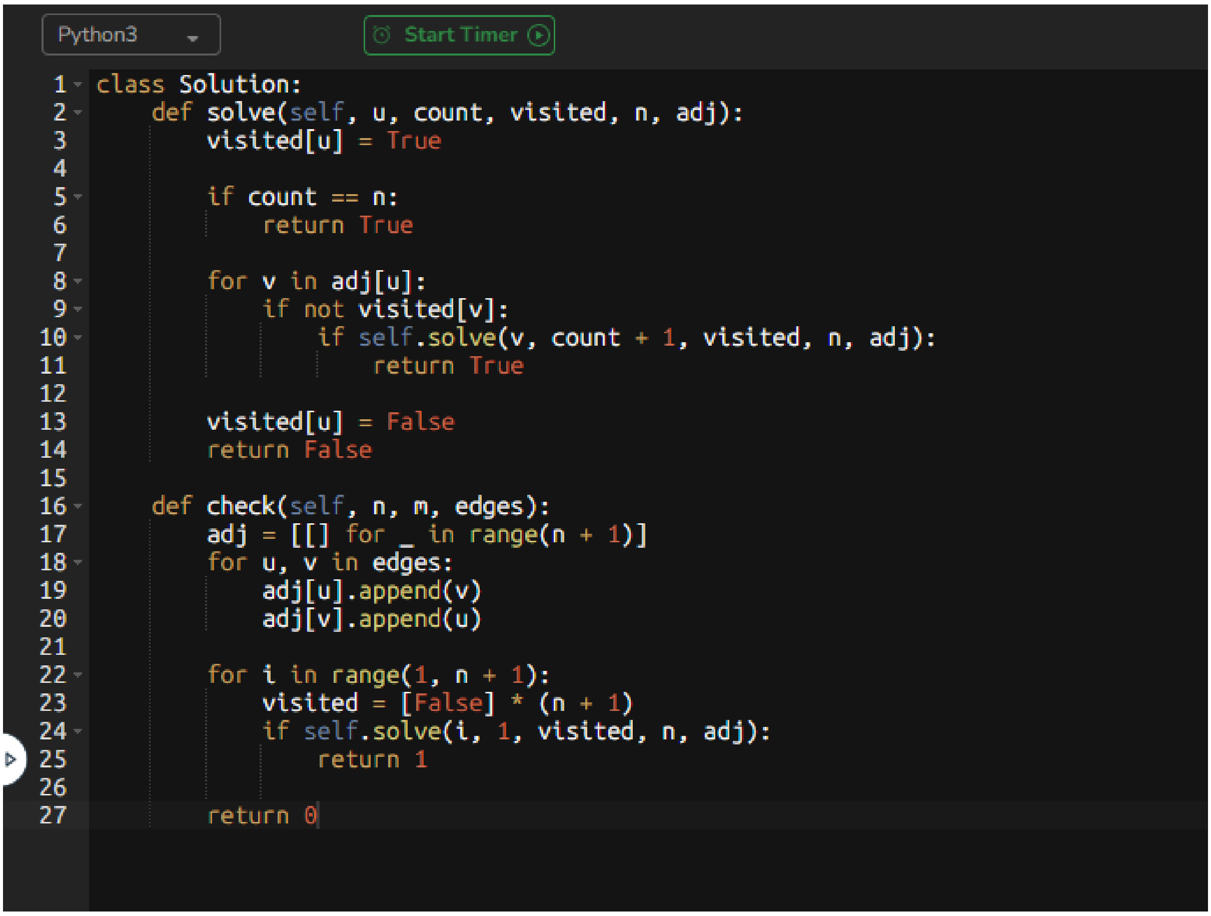
Constraints:

$1 \leq n \leq 10$

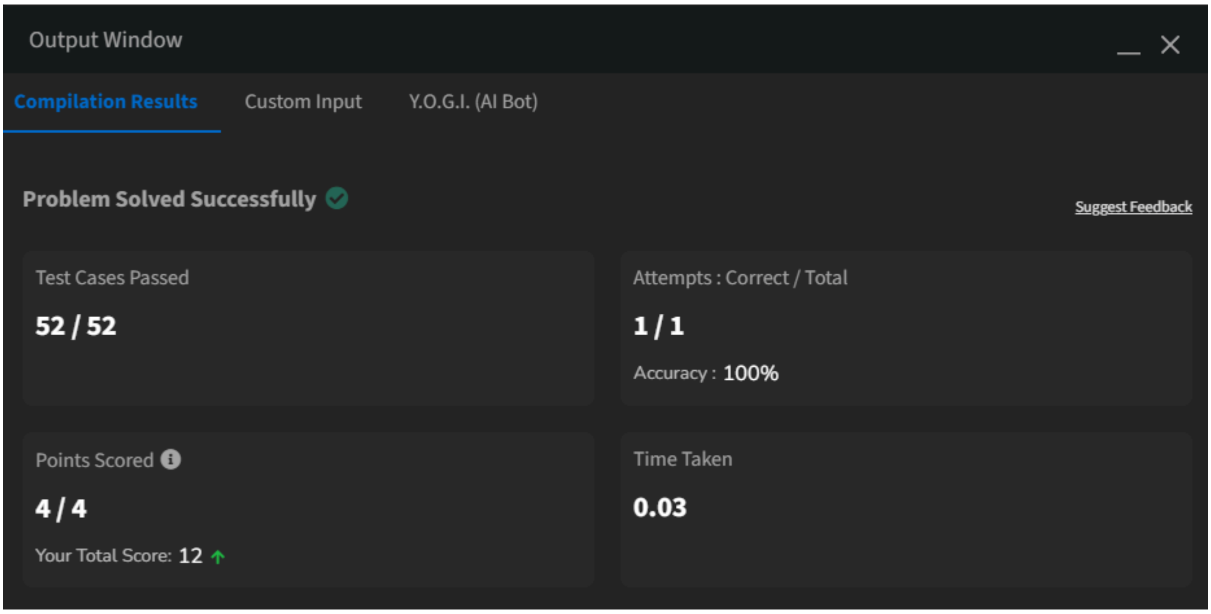
$1 \leq m \leq 15$

Size of edges[i] is 2

$1 \leq edges[i][0], edges[i][1] \leq n$



```python
class Solution:
    def solve(self, u, count, visited, n, adj):
        visited[u] = True

        if count == n:
            return True

        for v in adj[u]:
            if not visited[v]:
                if self.solve(v, count + 1, visited, n, adj):
                    return True

        visited[u] = False
        return False

    def check(self, n, m, edges):
        adj = [[] for _ in range(n + 1)]
        for u, v in edges:
            adj[u].append(v)
            adj[v].append(u)

        for i in range(1, n + 1):
            visited = [False] * (n + 1)
            if self.solve(i, 1, visited, n, adj):
                return 1

        return 0
```



Problem Solved Successfully

Test Cases Passed

52 / 52

Attempts : Correct / Total

1 / 1

Accuracy : 100%

Points Scored

4 / 4

Your Total Score: 12

Time Taken

0.03

Link : https://www.geeksforgeeks.org/problems/hamiltonian-path2522/1