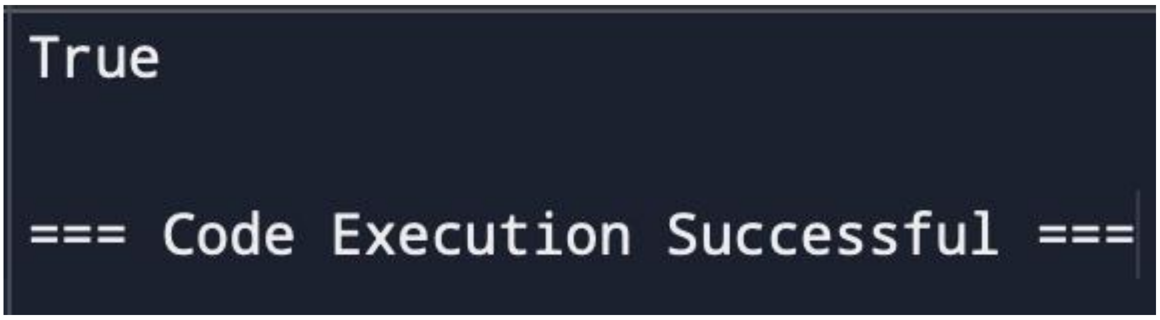
219. You are given an undirected graph represented by a list of edges and the number of vertices n. Your task is to determine if there exists a Hamiltonian cycle in the graph. A Hamiltonian cycle is a cycle that visits each vertex exactly once and returns to the starting vertex. Write a function that takes the list of edges and the number of vertices as input and returns true if there exists a Hamiltonian cycle in the graph, otherwise return false. Example:edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)] and n = 4

PROGRAM:-
```
def hamiltonian_cycle_exists(edges, n):
    graph = {i: set() for i in range(n)}
    for edge in edges:
        graph[edge[0]].add(edge[1])
        graph[edge[1]].add(edge[0])

    def dfs(node, visited, count):
        visited[node] = True
        count += 1
        if count == n:
            return True
        for neighbor in graph[node]:
            if not visited[neighbor]:
                if dfs(neighbor, visited, count):
                    return True
        visited[node] = False
        return False

    for start_node in range(n):
        visited = [False] * n
        if dfs(start_node, visited, 0):
            return True
    return False

# Example
edges = [(0, 1), (1, 2), (2, 3), (3, 0), (0, 2)]
n = 4
print(hamiltonian_cycle_exists(edges, n))  # Output: True
```
OUTPUT:-


```
True

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

TIME COMPLEXITY:- $O(n!)$