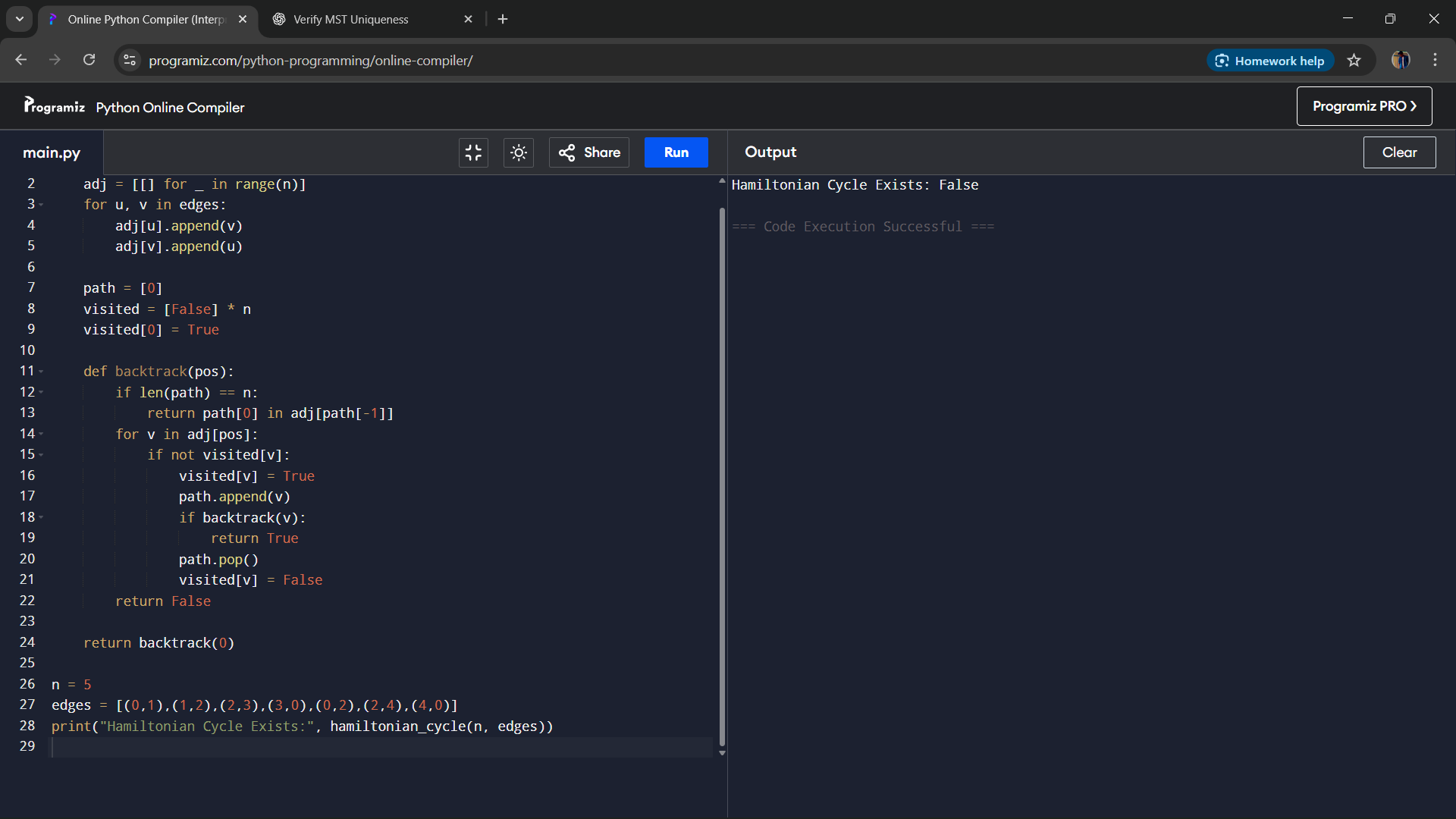
**Topic 6.12: Hamiltonian Cycle Detection**

**Question**  
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.

**Aim**  
To implement an algorithm that detects whether an undirected graph contains a Hamiltonian cycle.

**Algorithm**

1. Represent the graph using an adjacency list.
2. Start from vertex 0 and attempt to build a path visiting all vertices.
3. Use backtracking to explore possible paths:
   * Mark a vertex as visited and add it to the path.
   * Move to adjacent vertices that are not yet visited.
   * If all vertices are visited and there is an edge back to the starting vertex, a Hamiltonian cycle exists.
4. If no such cycle is found after exploring all possibilities, return false.

**Output**

**Result**  
The algorithm successfully determines whether a Hamiltonian cycle exists in an undirected graph and provides one possible cycle if it exists.

**Performance Analysis**

* Time Complexity: O(n!), since Hamiltonian cycle detection requires checking all permutations in the worst case.
* Space Complexity: O(n) for recursion and path storage.