Assignment no. 8

Title: Graph

- Part 1
    1. Graph Representation: Implement a program to represent a graph using adjacency matrix and adjacency list representations. Provide operations for adding edges and vertices.
    2. Create the following (using any graph representation of your choice)
        a. N (number of nodes in graph): Take input from user (should be >100)
        b. Add n vertices to the graph numbered from 1 to N
        c. Add an edge between two vertices x and y if x is divisible by y, for all 1<= x, y <= N
        d. Find the degrees of all vertices
        e. For any given vertex print all its neighbors
        f. For any given vertex print all the vertices that are neighbors of its neighbors (but not its direct neighbors)
        g. For any two/three given vertices print all their common neighbors
        h. Find the diameter of the graph
        i. Find all prime numbered vertices
- Part 2
    1. Depth-First Search (DFS): Write a program to perform depth-first search on a graph. Implement both recursive and iterative versions. Print the order in which nodes are visited.
    2. Breadth-First Search (BFS): Create a program to perform breadth-first search on a graph. Print the order in which nodes are visited.
    3. Shortest Path (Dijkstra's Algorithm): Implement Dijkstra's algorithm to find the shortest path from a source node to all other nodes in a weighted graph. Print the shortest distances.
- Part 3 *(Optional)*
    1. Minimum Spanning Tree (Prim's Algorithm): Write a program to find the minimum spanning tree of a connected, weighted graph using Prim's algorithm. Print the edges of the MST.
    2. Minimum Spanning Tree (Kruskal's Algorithm): Implement Kruskal's algorithm to find the minimum spanning tree of a connected, weighted graph. Print the edges of the MST.
    3. Flight Route Planner: Create a program that helps users plan flights between cities. Use a graph to represent routes and find the shortest path between two cities.
    4. Social Network Analysis: Design a program that analyzes a social network graph. Implement functions to find friends of friends, identify cliques, and detect influential nodes.
    5. Detect Cycle (Undirected Graph): Write a program to detect cycles in an undirected graph. Print whether a cycle is detected or not.
    6. Detect Cycle (Directed Graph): Implement a program to detect cycles in a directed graph. Print whether a cycle is detected or not.