EXPERIMENT 5

Objective: Implementing algorithms to find Euler tours in the graph.

Brief Theory:

A Euler tour in a graph is a closed path that visits every edge exactly once and returns to the starting vertex. Euler tours exist in Eulerian graphs, which satisfy specific conditions.

Key Concepts:

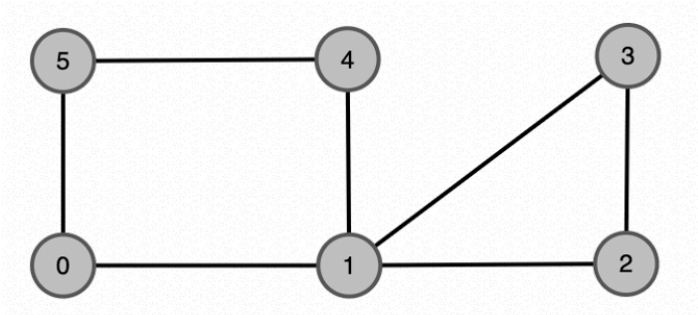
1. Eulerian Graph: A graph with a Euler tour must:
   - Be connected (all vertices are reachable).
   - Have all vertices with an even degree.
2. Semi-Eulerian Graph: If exactly two vertices have an odd degree, the graph contains an Euler path but not a closed Euler tour.

1. Initialize:
   - **Stack** ← empty stack.
   - **Circuit** ← empty list.
   - **CurrentVertex** ← any starting vertex.
2. While there are edges in the graph:
   - If **CurrentVertex** has unused edges:
     - Push **CurrentVertex** onto **Stack**.
     - Select and remove any edge (*CurrentVertex*, *v*).
     - Set **CurrentVertex** ← *v*.
   - Else:
     - Append **CurrentVertex** to **Circuit**.
     - Pop the top vertex from **Stack** (if not empty) and set it as **CurrentVertex**.
3. Append the final **CurrentVertex** to **Circuit**.
4. Return **Circuit** (in reverse order).

Fig. Hierholzer's Algorithm

Task:

1) Create a program that allows users to input a graph and check if it is Eulerian or Semi-Eulerian by verifying vertex degrees and graph connectivity. (Use Linked Representation)
2) Implement Hierholzer's algorithm to find and print a Euler tour for the given graph.



3) Display the Euler tour visually or as a sequence of edges.

Apparatus and components required: Computer with C or C++ Compiler and Linux/Windows platform.

Experimental/numerical procedure: Coding, compilation, editing, run and debugging.