Description:
The Seven Bridges of Konigsberg is a historically notable problem in mathematics. Its negative resolution by Leonhard Euler in 1736 laid the foundations of graph theory and prefigured the idea of topology. Euler shows that the possibility of a walk through a graph, traversing each edge exactly once, depends on the degrees of the nodes. The degree of a node is the number of edges touching it. Euler’s argument shows that a necessary condition for the walk of the desired form is that the graph be connected and have exactly zero or two nodes of odd degree.

Part 1 - Fill in the missing implementations in `WeightedGraph.java`.

```java
public boolean isEmpty()
// Returns true if this graph is empty; otherwise, returns false.

public boolean isFull()
// Returns true if this graph is full; otherwise, returns false.

public boolean hasVertex(T vertex)
// Returns true if this graph contains vertex; otherwise, returns false.
```

Part 2 - Compute whether or not the WeightedGraph contains a Eulerian path. Check the degree of a node by using a modified definition of the weight of an edge. We will define the weight of an edge to mean the number of edges that connect the two vertices. This can be computed by looking at the total weight of a row in the adjacency matrix. Use the conditions of a Eulerian path to determine if a Eulerian path exists or not. Create a method named `isEulerian` that returns true
or false. Fill in the missing implementations in WeightedGraph.java.

```java
public int vertexDegree(T vertex)
// Returns the degree of a vertex.

public boolean isEulerian()
// Returns true if this graph contains a Eulerian path; otherwise, returns false.
```

Rubric:
(10 points) Compiles without errors.
(10 points) Part 1 methods correctly implemented.
(10 points) Part 2 correctly implemented.

**Deliverables:** Submit on Blackboard.