CSC406Assignment#1: Due on or before February 11, 2015 rajasethupathy

Project: Implement the weighted/unweighted directed graph data structures using both adjacency matrix and adjacency list representation.

**Partial list of operations on a graph data structure:**

numNodes( ) – returns the number of nodes

numEdges( ) : returns the number of edges

existEdge( Edge e): returns true if e is an edge else returns false

existsEdge(int i, int j): returns true if there exists an edge between i and j else returns false

putEdge( Edge: e) : adds the edge e to the graph

putEdge( int i, int j) : adds the edge from i to j to the graph

removeEdge(Edge: e): deletes the edge e from the graph

removeEdge(int i, int j): deletes the edge from i to j from the graph

degree(Node: i): returns the degree of node i. this method is defined for undirected graphs only.

degree(int i): returns the degree of node i. this method is defined for undirected graphs only.

inDegree(Node: i): returns the in-degree of node i. this method is defined for directed graphs only.

inDegree(int: i): returns the in-degree of node i. this method is defined for directed graphs only.

outDegree(Node: i): returns the out-degree of node i. this method is defined for directed graphs only.

outDegree(int: i): returns the out-degree of node i. this method is defined for directed graphs only.

adjacentVertices(Node: i): returns the nodes that are adjacent to i

adjacentVertices(int: i): returns the nodes that are adjacent to i

areAdjacent(Node i, Node j): returns true if the nodes i and j are adjacent else returns false.

areAdjacent(int i, int j): returns true if the nodes i and j are adjacent else returns false.

General instructions:

1. Every graph has nodes and edges. So keep two variables, nodes and edges, in the root class. Keep adjusting these variables whenever you perform operations on the graph that warrant changes to these variables.
2. Implement fully as many methods as possible in the root class itself.

numNodes( ) and numEdges( ) are such methods.

1. You can also keep an array for outDegree in the root class. keep changing it whenever graph operations warrant changes to the outDegree of a node. Likewise for inDegree and degree.
2. You may assume that nodes are labeled by natural numbers.
3. Use an iterator to step thro elements of a collection.
4. Avoid repeating code.
5. Include your name, course#, semester, assignment#, date assigned and date due.
6. When using indices, don’t forget to do ‘range check’. A protected method?
7. File I/O is required.
8. Validation of input is not a requirement at this stage. However, it will be needed and required for future assignments.
9. Including a root class and a tester class, you are expected to submit six classes: G (root class), AMWDG, AMDG, ALWDG, ALDG, TesterClass.
10. Program has to be tested for at least one positive and one negative result.
11. Submit readable hard copy and flash-drive versions of the source code, input data, test data and a test run.
12. Seek immediate help if you need help in understanding/implementing your project.