Homework 6
Topological Sort

Due Wednesday December 6
Homework 6

Implement the methods addVertex, addEdge, topoSort and main in the TopologicalSort class shown on the following slides. The requirements for addVertex, addEdge and topoSort are shown in comments in the code. The main method will be used to test the addVertex, addEdge and topoSort methods. The main method expects one command line argument. The argument is the name of a text file. The main program should build the graph from the description in the file and then try to find a topological ordering for the graph. If a topological ordering exists, the main program should print the ordering to standard output. If a topological ordering does not exist, the main program should print the phrase "No topological ordering exists for the graph" to standard output.
Homework 6

The input file contains the description of a directed graph. The first line in the file is a sequence of vertex names separated by blanks. The remaining lines in the file represent edges. Each line contains two vertex names separated by a blank. The lines below show example contents of a file. Vertex names contain one or more letters.

A B C D E F G
A F
B F
B G
C A
C B
C D
C E
D A
E B
E D
F G
Homework 6

You will find it useful to implement some private methods to make your code more readable. For each private method include a brief comment indicating the purpose of the private method.

You must build the adjacency list representation of the graph as described by VertexNode, EdgeNode and the discussion in class.

To submit your solution send one file, TopologicalSort.java, to tgendreau@uwлax.edu.
import java.io.*;
import java.util.*;

public class TopologicalSort {
    // Adjacency list representation of a directed graph
    // See the class discussion for the details of the representation.
    private class VertexNode {
        private String name;
        private VertexNode nextV;
        private EdgeNode edges;
        private int indegree;

        private VertexNode(String n, VertexNode v) {
            name = n;
            nextV = v;
            edges = null;
            indegree = 0;
        }
    }
}
private class EdgeNode {
    private VertexNode vertex1;
    private VertexNode vertex2;
    private EdgeNode nextE;

    private EdgeNode(VertexNode v1, VertexNode v2, EdgeNode e) {
        vertex1 = v1;
        vertex2 = v2;
        nextE = e;
    }
}

private VertexNode vertices;
private int numVertices;
Homework 6

public TopologicalSort() {
    vertices = null;
    numVertices = 0;
}

public void addVertex(String s) {
    //PRE: the vertex list is sorted in ascending order using the name as the key
    //POST: a vertex with name s as been add to the vertex list and the vertex list
    //       is sorted in ascending order using the name as the key
}
public void addEdge(String n1, String n2) {
    //PRE: the vertices n1 and n2 have already been added
    //POST: the new edge (n1, n2) has been added to the n1 edge list
}

public String topoSort() {
    //if the graph contains a cycle return null
    //otherwise return a string containing the names of vertices separated by blanks
    //in topological order.
}

public static void main(String args[]) throws IOException{
    //see problem statement
}
}