Put each of the problems on a separate sheet of paper, and make sure your name is on each sheet. To hand them in, staple them together and bring them to the class on due date. Please make sure to express your algorithms in pseudo-code when directed (see the appropriate section of the course textbook for the proper pseudo-code style), and always provide justification for your answer when asked to give the running time of an algorithm. Be brief and concise, and draw pictures where appropriate.

**Problem 1.**

(a) Would you prefer DFS or BFS (or both equally) for the following tasks? (Assume the graph is undirected and connected.) Justify your answer.

1. Determining if the graph is acyclic
2. Finding a path to a vertex known to be near the starting vertex
3. Finding the connected components of the graph

(b) A graph is **triconnected** if one has two remove at least 3 vertices from the graph to disconnect it. Construct examples of the following graphs or explain why it cannot be done.

1. A triconnected graph with exactly 5 vertices and 8 edges
2. A triconnected graph with exactly 5 vertices and 6 edges
3. A triconnected graph with exactly 8 vertices and 14 edges

**Problem 2.** A company named RT&T has a network of $n$ switching stations connected by $m$ high-speed communication links. Each customer’s phone is directly connected to one station in his or her area. The engineers of RT&T have developed a prototype video-phone system that allows two customers to see each other during a phone call. In order to have acceptable image quality, however, the number of links used to transmit video signals between the two parties cannot exceed 4. Suppose that RT&T network network is represented by a graph. Design an efficient algorithm that computes,
for each station, the set of stations it can reach using no more then 4 links. Analyze its running time.

**Problem 3.** An Euler tour of a directed graph G with \( n \) vertices and \( m \) edges is a cycle that traverses each edge of G exactly once according to its direction. Such a tour always exists if the in-degree is equal to the out-degree for each vertex in G. Describe an \( O(n + m) \) time algorithm for finding a Euler tour of such a graph G. Analyze its running time.

**Problem 4.** Trace the execution of TopologicalSort algorithm on the following graph. To review, here is the pseudo-code for the algorithm:

```
Algorithm TopologicalSort(G)
Input: A digraph G with \( n \) vertices
Output: A topological ordering \( v_1, v_2, \ldots, v_n \) of G
Let Q be an initially empty queue
for each vertex u in G do
    Let incounter(u) be the in-degree of u
    if incounter(u) = 0 then
        Q.enqueue(u)
    end if
end for
i ← 1
while (!Q.isEmpty()) do
    u ← Q.dequeue()
    Let u be vertex number i in the topological ordering.
    i ← i + 1
    for each outgoing edge \((u, w)\) of u do
        incounter(w) ← incounter(w) − 1
        if incounter(w) = 0 then
            Q.enqueue(w)
        end if
    end for
end while
```

Show the graph after each iteration of the second while loop, and display
the encounter and the currently assigned topological sorting labels at each one of these iterations.

**Problem 5.** Bob loves foreign languages and wants to plan his course schedule to take the following nine language courses: LA15, LA16, LA22, LA31, LA32, LA126, LA127, LA141, and LA169. The course prerequisites are:

- LA15: (none)
- LA16: LA15
- LA22: (none)
- LA31: LA15
- LA32: LA16, LA31
- LA126: LA22, LA32
- LA127: LA16
- LA141: LA22, LA16
- LA169: LA32

Find a sequence of courses that allows Bob to satisfy all the prerequisites.