1. Give an algorithm to detect whether a given undirected graph contains a cycle. If the graph contains a cycle, then your algorithm should output one. (It should not output all cycles in the graph, just one of them). The running time of your algorithm should be $O(m + n)$ for a graph with $n$ nodes and $m$ edges.

2. Given an undirected graph $G = (V, E)$ and an integer $k$, find an induced subgraph $H = (U, F)$ of $G$ of maximum size (maximum in terms of the number of vertices) such that all vertices of $H$ have degree at least $k$, i.e., each vertex in $H$ has at least $k$ neighbors in $H$.

3. We have a connected graph $G = (V, E)$, and a specific vertex $u \in V$. Suppose we compute a depth-first search tree rooted at $u$, and obtain a tree $T$ that includes all nodes of $G$. Suppose we then compute breadth-first search tree rooted at $u$, and obtain the same tree $T$. Prove that $G = T$. (In other words, if $T$ is both a depth-first search tree and a breadth-first search tree rooted at $u$, then $G$ cannot contain any edges that do not belong to $T$.

4. Prove or disprove: If a directed graph $G$ contains cycles, then the topological sort algorithm done in class produces a vertex ordering that minimizes the number of “bad” edges that are inconsistent with the ordering produced.