1. True or False: The reverse postorder of a graph's reverse is the same as the post-order of the graph. How is this question relevant to Kosaraju's algorithm?

2. Prove that every connected undirected graph has a vertex whose removal (including all adjacent edges) will not disconnect the graph.

3. Given a connected undirected graph, design an algorithm to find a vertex whose removal will not disconnect the graph.

4. Given a directed graph with V vertices and E edges, design an efficient algorithm to find a directed cycle with the **minimum** number of edges (or report that the graph is acyclic).

   Your algorithm should run in $O(EV)$ time and $O(E+V)$ space. Assume $V \leq E \leq V^2$.

5. Given an directed graph and a starting vertex u, give an algorithm for finding all vertices such that there is an odd-length path to those vertices from u. These paths may involve cycles. Your algorithm should complete in $O(E+V)$ time. If you're stuck, you might also try to come up with an algorithm that completes in $O(EV)$ time.

6. In class we implemented DFS using the simple recursive routine shown below.

```
private void dfs(Digraph G, int v) {
    marked[v] = true;
    for (int w : G.adj(v)) {
        if (!marked[w]) dfs(G, w);
    }
}
```

   By contrast, we implemented BFS using the iterative code shown below:

```
while (!q.isEmpty()) {
    int v = q.dequeue();
    for (int w : G.adj(v)) {
        if (!marked[w]) {
            edgeTo[w] = v;
            distTo[w] = distTo[v] + 1;
            marked[w] = true;
            q.enqueue(w);
        }
    }
}
```

   Suppose we take the BFS code and replace the Queue with a Stack. Is the resulting search BFS? DFS? Something else?

7. Extra challenging: An Eulerian cycle is a directed cycle that contains each edge exactly once. Design an algorithm that finds an Eulerian cycle or reports that no such tour exists.