1. Hand-simulate the execution of the strongly connected components algorithm given in lecture on the graph shown in Figure 23.6 (page 484). For definiteness, start at vertex \( r \), and whenever there is a choice of which edge to explore next, take the edge to the alphabetically least neighbor among the remaining unexplored edges. (Note that this is not a requirement of the algorithm; it’s just to make Sung’s life easier when she grades them.) Please give a fairly detailed trace of the actions of the algorithm, showing the order in which vertices/edges are examined, stack pushes/pops, classification of edges as tree-, back-, etc. E.g., you trace should start with something like:

   Visit \( r \); push it; assign dfs number 1; initialize \( r.LOW \) to 1.
   Visit \( u \); push it; assign dfs number 2; initialize \( u.LOW \) to 2; \((r,u)\) is a tree-edge.
   ...

List the dfs number assigned to each vertex, together with the root of its component, all exits, LOW values, and the component number to which it is assigned. Draw the dfs tree in the way we’ve been doing in examples in class: children of any node appear left to right in the order they are visited, etc. Circle each SCC.

2. The SCC algorithm I gave is in fact incomplete in that I never said what to do if dfs doesn’t visit all vertices from its starting vertex. E.g., in the above example if you start dfs from \( z \), you’ll only find \( z \) and \( x \). Propose a fix, i.e. a simple “top level” routine to add that calls the SCC subroutine I gave, that will find all components, and still operate in linear time. Briefly outline how it would work on the above example in case \( r \) happens not to be the start vertex. Explain informally why your method is correct.

3. As part of the correctness proof for the SCC algorithm, I proved that every vertex in a given SCC will be a descendant of the SCC’s root in the dfs tree. Prove the following stronger property: the vertices in an SCC will be contiguous descendants of the root, by which I mean that for each vertex in a given SCC, all of its ancestors up to and including its root will also be in that SCC.

4. Text page 494, exercise 23.5-7.

5. Text page 599, exercise 27.2-3.