Question 1 (Connected Components, 30 points). Identify the labels of the vertices in the strongly connected components of the graph below.

The components are \{A, C, E, F\}, \{B, D, G, H, I\}, \{J\}. First, we see that these are strongly connected. The first component is strongly connected because we have cycles ACF and AEF. The second is strongly connected because of the cycle BIHGD. The last one is only a single vertex, and thus connected. To show that these components are not connected to each other, note that no edges connect into the second component or out of the first component from anywhere else.
Question 2 (Cycles Through a Vertex, 35 points). Find a linear time algorithm that given a directed graph $G$ and a vertex, $w$, determines whether or not $G$ contains a cycle through $w$, and show that your algorithm is correct.

The algorithm is as follows:

Set all vertices as unvisited
Explore($w$)
For each vertex $u$
  If $u$ is visited and there is an edge from $u$ to $w$
    Return 'There exists a cycle'
  
Return 'There does not exist a cycle'

This works because there is a cycle through $w$ if and only if there is some vertex, $u$, reachable from $w$, with an edge to $w$. Running the explore routine above sets exactly those $u$ reachable from $w$ as visited, and we then check to see if any have an edge to $w$. The runtime of the algorithm is $O(|V|)$ to set the vertices as unvisited, $O(|V| + |E|)$ to run the explore step, and $O(|V|)$ to run the final loop over $u$'s. Thus, the total runtime is $O(|V| + |E|)$. 

Question 3 (Minimax Path, 35 points). Modify Dijkstra’s algorithm so that given a weighted graph $G$ and two vertices, $s$ and $t$, it computes the minimum value $x$ so that there exists an $s-t$ path all of whose edges have weight at most $x$. You do not need to justify your answer.

The algorithm is as follows:

Set $\text{dist}(u) = \infty$ for all vertices $u$
Set $\text{dist}(s) = 0$
Let $Q$ be a priority queue
Insert all vertices into $Q$ using $\text{dist}$ as keys
While $Q$ is not empty
    $u = \text{eject}(Q)$
    For $w$ adjacent to $u$
        If $\max(\text{dist}(u), \ell(u, w)) < \text{dist}(w)$
            Set $\text{dist}(w) = \max(\text{dist}(u), \ell(u, w))$
            $\text{DecreaseKey}(w)$
Return $\text{dist}(t)$

[To show correctness, we claim that at the start of each iteration of the while loop, $\text{dist}(u)$ is the minimum value of $x$ so that there is an $s-u$ path consisting of edges of weight at most $x$ that passes only through nodes that have been ejected from $Q$ (or infinity if no such path exists). Furthermore, once $u$ has been ejected, we claim that this is actually the minimum value of $x$ over all such paths. These are both clearly true when $Q$ is initialized, so we merely need to show that they are maintained.

When $u$ is ejected, we note that since it has the smallest remaining key, that any path that leaves the set of ejected vertices must contain an edge of length at least $\text{dist}(u)$. Therefore, $\text{dist}(u)$ is actually the minimum over all $s-u$ paths. Furthermore, for any $w$, the minimum max edge weight of $s-w$ paths only though ejected edges is the smaller of the old minimum (i.e. the old $\text{dist}(w)$) and the minimum over paths that pass through $u$, which would be $\max(\text{dist}(u), \ell(u, w))$. Thus, after updating $\text{dist}(w)$, the invariant is maintained.]