Solution to Homework 1
Group members' netid: rjones27, jharris39, khan14

1. **Problem 1: The Closet Mayhem Problem**
   (A) Don't Have!

   (B) Don't Have!

   (C) Don't Have!

   (D) Don't Have!

   (E) Assuming X is a strongly connected component of Graph G and X is a sink then X' (the mirror X) is not a source.
   Given X has no outgoing edges because it is a sink and X' has at least 1 incoming edge let $[i : x_i]$ and $[i : x_i']$ be two complementing nodes for the ith broom outside of X and X'. Now let $[i : X_i]$ force some node in X'. Therefore by the proof done in problem B we know X must have a edge between the node $[i : x_i']$ and the complement of node $[i : x_i]$ and X' to each other.
   This outgoing edge from X to $[i : x_i']$ makes X not a sink and therefore contradicts the hypothesis. So if X is a sink the X' has to be a source.

   (F) Don't Have!

2. **Problem 2: Heavy Time**

(A) Check each node if it is a sink if so then add an edge to the target sink:

```
for each node in the adjacency list do
    if # of out going edges = 0 then
        Add edge to $s$
    end if
end for
```

Running Time: $O(n)$
Number of Edges Worst Case: m

(B) First, reverse the graph and topological sort the the vertices. Then, use DFS to find all the vertices reachable from s and remove the vertices that are not.

```
for each vertex v do
    $v.t_max = 0$
end for
for each vertex v in the topological order do
    if v is significant then
        $v.t_{max} = t_{max} + 1$
    end if
    for each edge (v,u) do
        if $v.t_{max} > u.t_{max}$ then
            $u.t_{max} = v.t_{max}$
        end if
    end for
end for
```

Then, any vertex with a $t_{max} > t$ can reach s via a path going through t significant vertices.

(C) Our algorithm is to negate all the weights on the edges, then use the Bellman-Ford algorithm. Let G' be G with all the edge weights negated. Since there are no cycles in G, there will be no negative cycles; therefore, the Bellman-Ford algorithm will find the shortest distance between v to s in our modified graph, G'. The weight of every path will be the opposite of what it was for G, so the lightest path in G' will be the heaviest in G. Negating every edge weight require $O(m)$ time, and the Bellman-Ford algorithm takes $O(nm)$, so this algorithm runs in $O(nm)$.

(D) First, topological sort the the vertices. If there is a path that visits all of the vertices, it will be the only topological sorting.
Proof: Suppose there are multiple topological sortings. There must be at least two vertices that could be in a different order. Let A and B be two such vertices. Then there must be no path from A to B nor from B to A. If there was a path

from A to B, then A would have to come before B in the sort order. If there was a path from B to A, then B would have to come before A in the sort order. So, there is no path that goes through both A and B, so there is no path that visits every vertex.
To find the path that visits all vertices, start at the source (first in the topological sort). Then, from each node, go to the next one in the topological sorting. The edges traversed will form the path that visits all the vertices, if it exists.

3. **Wishful Graph**

   (A) R is an equivalence relation. R is defined as uRv iff (u can reach v) or (v can reach u) Clearly, u and v can be interchanged thus it is symmetric.
   If aRb and bRc then a can reach b and b can reach c by the definition. So then a must be able to reach c so aRc given aRb and bRc, therefore R is transitive.
   Since a node can always reach itself we know that aRa therefore R is reflexive.
   Since R is symmetric, transitive, and reflexive we know R is an equivalence relation.

   (B) Our algorithm was to first run the **Undirected Simple SCC** on G. Then run a **Topological Sort** on G which should only work if it is a DAG and thus not uselessly connected. If it is false then it is uselessly connected. (Note: since both Algorithms are defined in class we did not write the algorithms).