**Question 1** (Topological Sort, 30 points). *Provide a linear ordering of the DAG given below. In particular, list the letters A through J in an order consistent with the graph provided. You do not need to explain how you found it.*

Applying DFS on $G$, we find that the pre- and post- visits in order are:

- $\text{pre} - A, \text{post} - A, \text{pre} - B, \text{pre} - C, \text{post} - C, \text{post} - B, \text{pre} - D, \text{pre} - I, \text{pre} - H, \text{post} - H,$
- $\text{post} - I, \text{post} - D, \text{pre} - E, \text{post} - E, \text{pre} - F, \text{pre} - G, \text{post} - G, \text{post} - F, \text{pre} - J, \text{post} - J.$

Putting the vertices in reverse-postorder, we obtain


Note: There are many other acceptable answers.
**Question 2** (Connectivity and Pre/Post-Orderings, 35 points). Let $G$ be an undirected graph. Show that $G$ is connected if and only if when performing a depth first search on $G$ that the vertex with the smallest preorder number is the same as the vertex with the largest postorder number.

Let $v$ be the first vertex explored in the DFS search of $G$. Notice that $G$ is connected if and only if $\text{explore}(v)$ discovers all vertices in the graph. Notice that $v$ is the first visited vertex, and thus has the smallest previsit number. On the other hand $v$'s postvisit number is larger than the postvisit number of $w$ if and only if $w$ was discovered while exploring $v$. Thus, the same vertex has both the smallest previsit and largest postvisit numbers if and only if all vertices are discovered while exploring $v$, and thus happens if and only if $G$ is connected.
**Question 3** (Subway Navigation, 35 points). A map of a subway system is given by an undirected graph $G$. In addition, each edge of this graph is given a color to designate which of several subway lines that edge is a part of. Give a **linear time** algorithm that given two vertices of this graph returns the smallest possible number of times that one would need to change lines in order to get from one vertex to the other. You may assume that all the edges in a given line are connected. In particular, it should be possible to get from any station on one line to any other on that line without changing lines.

Partial credit will be given for any correct, polynomial-time algorithm. You do not need to prove correctness or runtime.

We think of a new problem. First, starting at the source station, $s = s_0$, we get on some line $\ell_1$. We travel along this line and get off at station $s_1$. From there we transfer to line $\ell_2$, which we take to station $s_2$ and so on. Eventually, we end up at station $s_k$, which is our destination. We want to minimize $k$, the number of lines used in such a route. Note though that the number of steps on this path of transferring between stations and lines has length $2k$. Therefore, we can write this problem as that of finding shortest paths in a different graph $G'$. Our pseudocode is as follows:

1. Create a new graph $G'$
2. The vertices of $G'$ are given by either:
   - A color, or line on our subway map.
   - A station on our subway map.
3. For each edge in $G$, add an edge to $G'$ between the line the edge is a part of and the stations on either end of it.
4. Let $v$ be our starting vertex and $w$ our destination vertex, both correspond to station-vertices of $G'$.
5. Run BFS($G'$, $v$).
6. Return $\text{dist}(w)/2 - 1$. (We want the number of times we change lines, which is one less than the total number of lines used)

To analyze the runtime of this algorithm, we note that the number of vertices of $G'$ is the number of vertices of $G$ plus the number of lines, which is at most $|V| + |E|$. Furthermore, we can produce $G'$ by creating a vertex for each station and line, and then for each edge of $G$ creating an edge between the appropriate line and the two stations on either side. Thus, we can produce the graph $G'$ in $O(|V| + |E|)$ time. Finally, the Breadth First Search runs in time $O(|V'| + |E'|) = O(|V| + |E|)$. Thus, our entire algorithm runs in time $O(|V| + |E|)$. 
