1. (a) We iterate through the graph and count (edges (prerequisites) each node has. Then we DFS on the node with no incoming edges, adding them to output list in order they are visited. If it has cycle it outputs "impossible".

(b) Same as (a) but instead uses BFS to find ~~tasks~~ courses with no prerequisites, then iterate through the tasks. Prints "impossible" if cycle is found.

2. Same as 1(b) uses modified BFS to provide lexicographically smallest valid course sequence. We actually sort the neighbour nodes in ascending order to achieve that.

3. We run DFS for finishing order. We run DFS on ~~the~~ reversed graph using the previously found finishing order. Thus each reachable group of nodes in the step ~~that~~ is found forms a strongly connected component.