Name: Md. Mehedi Hasan Tanvir ID: 2210107

Explanation Lab 5 CSE 221

#Task 1A: Here I've taken a list called indegL, that counts and store the indegree of all the nodes. Then I take that list and form nodes with 0 indegree I form another queue. Then I call DFS on the values of queue one by one. If after all these the visited list is of the same size as the adj list of the graph, then we can say It's possible to take all courses in the order of DFS, that is stored in visited list.

#Task 1B: This also follows pretty much the same process, except here we run BFS instead of DFS. Something I forgot to add on to task1a's explanation is on the DFS/BFS part we have to have to check if our current node is the only indegree left to go through before we move on to it's child node. If not, then, we decrease 1 from indeg list and leave it so another parent can get us through it.

#Task 2 : Here, just like task1A, we run DFS with consideration for each nodes indegree. But here on the step of DFS where we get to a node's outdegree list we create another list that is sorted. Then traverse that instead of the outdegree list. Additionally we also form the queue and sort it before calling DFS. This ensures lexicographical order as desired.

#Task 3: Here I used kosarajus algorithm to find the strongly connected components. First with the help of indegree list I tried to find the best possible node to start DFS from. Then we run the DFS and keep a list as stack to keep track of what nodes finish first. We reverse it later on to get our desired stack. We then ~~reverse~~ transpose the graph or reverse the edge directions. Then we perform DFS on the first value of stack. then we pop it, At the same time putting the dfs route in the output. If first value is already in visited we simply pop it. This continues till stack is empty.