Task-1:

id: 21301289

In this problem, we were tasked with finding an order in which courses can be completed based on their pre requisite requirment. We implement two approaches: DFS approach and BFS approach.

The DFS approach used to recursive function to explore the courses in a depht first manner, and it utilized a stack to keep track of the order in which courses were completed. The DFS aproch is a topological sorting algo and it produces a valid order if there is one. if there is no valid, it print "IMPOSSIBLE".

The BFS approach on the other hand, used a queue to explore the courses in a breath first manner. It calculated the in-degree of each course and iteratively removed courses with an in-degree of zero while updating the in-degree of their neighbors. The BFS approach, like the DFS approach, also result in a valid order if one exist, and it print "Impossible" Otherwise.

Both approach have a time complexity of O(N+M) where N is the number of courses and M is the number of prerequisite requirements.

Task-3:

id: 21301289

The program reads the input, which consists of the number of edges (M) and vertices (N), as well as the directed edges that connect the vertices.
Each vertex is represented as a key and the list associated with each key comprises its neighbours.
The initialises an empty graph as an adjacency list.
The DFS method is then used on the original graph to fill the stack. Using the vertex finishing times from the DFS traverse, the stack is filled.

After obtaining the stack with finishing times, the program proceeds to find the strongly connected components, it performs another DFS traversal on the reversed graph using the stacks ordering. Each DFS call identifies a SCC and the vertices belonging to that components are collected in a list.

Finally, program prints the SCC, where each component is printed on a separate line, and the vertices within the component are separated by spaces.