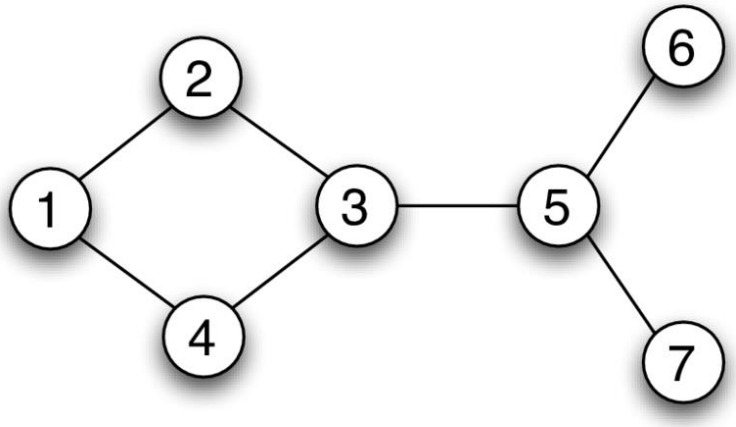
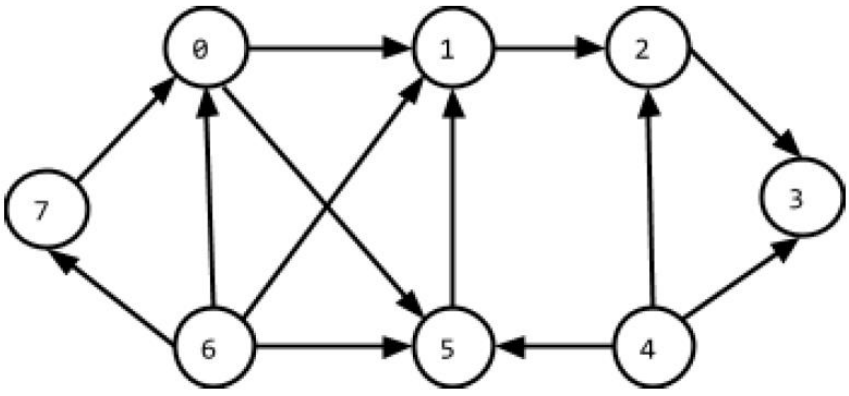
1. Write a function ***findCommonAdjacentNodes*** that takes two vertices A and B and shows the common adjacent nodes of A and B. For example: In this given graph, ***findCommonAdjacentNodes (2, 5)*** will return 3.



2. Consider a directed graph G with n nodes. Write a function ***findUnreachableNode*** that takes a node and prints all the nodes that are unreachable from the given node. You can use either adjacency list or adjacency matrix to solve this problem.
Function Signature: ***findUnreachableNode (node)***
For example: In the following graph, ***findUnreachableNode (0)*** will return 4, 6, 7 as they are unreachable from node 0.



3. Consider this same graph (Problem 2) G with 8 nodes. Write a code to add these nodes and edges (Not user input. You can manually add the edges in you code) so that this code can represent this graph using adjacency matrix.

(a) Using this adjacency matrix, write code to find and print the sink nodes of this given graph. Sink node is the node that has no outgoing edges. It can have any number of incoming edges.
Here, Node 3 is the sink node since no edges are coming out of it.

(b) Using this adjacency matrix, write code to find and print the node with the highest outdegree.
Here, Node 6 has the highest outdegree which is 4.

4. Given a source node source, find all the path that can be traversed to reach node destination.
For example, if source is 0 and destination is 8 - the paths will be:
0 1 8
0 8
0 7 5 8
0 7 6 5 8
If source is 0 and destination is 4 - the paths will be:
0 1 8 3 2 4
0 8 3 2 4
0 7 5 8 3 2 4
0 7 5 3 2 4
0 7 6 5 3 2 4
0 7 6 5 4
0 7 5 4
0 7 6 5 8 3 2 4
The function signature is:
printPath(Graph, source, destination)