1. Discuss Homework-00 and comprehension quizzes in case required.

2. Consider the following recursive function:

\[
F(n) = \begin{array}{ll}
\text{if } n > 1 & F(\lfloor n/3 \rfloor) \\
& \text{Print(“Hello World”)}
\end{array}
\]

Let \( R(n) \) denote the number of times this function prints “Hello World” given the positive integer \( n \) as input. Which of the following are true? Give reasons.

A. \( R(n) = \mathcal{O}(n) \)
B. \( R(n) = \Omega(\log_2 n) \)
C. \( R(n) = \lceil \log_3 n \rceil + 1 \)
D. \( R(n) = \lfloor \log_3 n \rfloor + 1 \)

3. Prove or disprove: There are an even number of odd-degree vertices in any undirected graph.

4. In the lectures, we learnt that the two main graph representations are adjacency list and adjacency matrix. Design algorithms for the following tasks:

(a) Given a graph \( G = (V, E) \) as input in the adjacency matrix representation, output the adjacency list representation of \( G \).

(b) Given a graph \( G = (V, E) \) as input in the adjacency list representation, output the adjacency matrix representation of \( G \).

Give pseudocode and discuss running time of the your algorithms.

5. The reverse of a directed graph \( G = (V, E) \) is another directed graph \( G^R = (V, E^R) \) on the same vertex set but with all the edges reversed. Design an algorithm that outputs the adjacency list of the reverse of a given graph \( G \). \( G \) given as input in adjacency list format. Discuss running time.

6. While discussing the \texttt{explore}(G, v) procedure in the lectures, we noticed that we obtain a rooted tree that contains all nodes that are reachable from \( v \) in \( G \). Moreover, the edges in the tree are exactly the edges that were traversed during exploration. (Recall that a rooted tree is a directed graph where there is a directed edge from a parent of a node to itself.)

For example, when we executed \texttt{explore}(G, a) for the graph below (on the left), we obtained the following rooted tree (on the right).
Let us formally define this rooted tree. The root of this tree is the node on which the explore procedure is first called and a node $u$ is the parent of node $v$ iff $u$ caused the immediate discovery of $v$ (that is, $v$ was explored while looking at the neighbors of $u$). Let us call this tree a DFS tree. Note that the DFS tree depends on the order in which neighbors of a vertex are considered. That is, if you change the order of vertices in the linked list corresponding to a vertex, then you may obtain a different DFS tree, though containing the same set of nodes.

Design an algorithm that returns the adjacency list of the DFS tree obtained on executing the explore procedure.