S.No	Question	Marks
1.	Solve the following recurrence relation using Masters theorem. T(n) = 4 T(n/2) + n	
2.	Given the head of a linked list, determine if there is a cycle in it using **Floyd's cycle detection algorithm**.	
3.	Given an array of integers and a target value, return the indices of the **two numbers** that add up to the target. Input: [2,7,11,15], target = 9 → Output: [0,1]	
4.	Given an array, print the **next greater element** for every element in the array. If no greater element exists, print -1	
5.	Design an algorithm to find the index of the target timestamp in O(log n) time	
6.	Given a binary tree, print its **inorder, preorder, and postorder** traversals (both recursive and iterative methods)	
7.	**Detect a cycle** in a linked list (Floyd's Cycle Detection)	
8.	The **diameter** is the length of the longest path between any two nodes. Find the diameter of a binary tree in O(n) time	
9.	Given an undirected graph, print **Breadth-First Search (BFS)** and **Depth-First Search (DFS)** traversals starting from a source node.	
10.	Given the head of a linked list, determine if there is a cycle in it using **Floyd's cycle detection algorithm**.	
11.	Implement a queue with the standard enqueue() and dequeue() operations using only two stacks.	
12.	Given a weighted graph and a source node, find the **shortest path** to all other nodes using Dijkstra's algorithm.	
13.	Implement a queue with the standard enqueue() and dequeue() operations using only two stacks.	
14.	**Detect a cycle** in a linked list (Floyd's Cycle Detection)	
15.	Given a directed graph, check whether it contains a cycle using **DFS**	
16.	Given an array of integers and a target value, return the indices of the **two**	

	numbers that add up to the target. Input: [2,7,11,15], target = 9 → Output: [0,1]	
17.	Given the head of a linked list, determine if there is a cycle in it using **Floyd's cycle detection algorithm**.	
18.	Implement a queue with the standard enqueue() and dequeue() operations using only two stacks.	
19.	Given an array, print the **next greater element** for every element in the array. If no greater element exists, print -1	
20.	Implement a queue with the standard enqueue() and dequeue() operations using only two stacks.	
21.	Given an array, print the **next greater element** for every element in the array. If no greater element exists, print -1	
22.	Given the head of a linked list, determine if there is a cycle in it using **Floyd's cycle detection algorithm**.	
23.	Implement a queue with the standard enqueue() and dequeue() operations using only two stacks.	
24.	**Detect a cycle** in a linked list (Floyd's Cycle Detection)	
25.	Implement a queue with the standard enqueue() and dequeue() operations using only two stacks.	
26.	Given a string, find the **length of the longest substring** without repeating characters. Example: "abcabcbb" → Output: 3 ("abc")	
27.	Given an array, print the **next greater element** for every element in the array. If no greater element exists, print -1	
28.	Design an algorithm to find the index of the target timestamp in O(log n) time	
29.	Given a string, find the **length of the longest substring** without repeating characters. Example: "abcabcbb" → Output: 3 ("abc")	
30.	Implement a queue with the standard enqueue() and dequeue() operations using only two stacks.	
31.	Given an array of integers and a target value, return the indices of the **two**	

	numbers that add up to the target. Input: [2,7,11,15], target = 9 → Output: [0,1]	