Q1. Define data structure. Give examples. (3 marks)
Ans: A data structure is a way of organizing and storing data so that it can be accessed and modified efficiently. Examples include arrays, linked lists, stacks, queues, trees, and graphs.

Q2. What is the time complexity of binary search in a sorted array? (2 marks)
Ans: $O(\log n)$

Q3. Choose the correct options: (3 marks)
(i) Which of the following is a linear data structure? (1 marks)
A. Array B. Binary Tree C. Graph D. Hash Table
Ans (i): A

(ii) Which data structure uses LIFO principle? (1 marks)
A. Queue B. Tree C. Stack D. Linked List
Ans (ii): C

(iii) Which traversal is used in Depth-First Search? (1 marks)
A. Level Order B. Preorder C. Breadth First D. Postorder
Ans (iii): B

Q4. Differentiate between Stack and Queue. (3 marks)
Ans: Stack follows LIFO (Last In First Out) while Queue follows FIFO (First In First Out). In stack, elements are inserted and deleted from the same end, whereas in a queue, insertion is at the rear and deletion is from the front.

Q5. Answer the following: (3 marks)

(i) What is a circular linked list? (1 marks)
Ans (i): A linked list in which the last node points back to the first node.

(ii) What is the maximum number of children a binary tree node can have? (1 marks)
Ans (ii): Two

(iii) What is the worst-case time complexity of Quick Sort? (1 marks)
Ans (iii): $O(n^2)$

Q6. Explain the operations on a queue with suitable examples. (2 marks)
Ans: A queue supports enqueue (insert), dequeue (remove), peek (front element), and isEmpty operations. It follows FIFO order.

Q7. Fill in the blanks: (2 marks)
(i) A binary tree with all levels filled except possibly the last, which is filled from left to right, is called a ________. (1 marks)
Ans (i): Complete Binary Tree

(ii) The minimum number of nodes in an AVL tree of height 3 is ________. (1 marks)
Ans (ii): 4

Q8. Explain the concept of a stack and its applications. (5 marks)
Ans: A stack is a linear data structure that follows the Last In, First Out (LIFO) principle, where elements are inserted and removed only from one end, called the top of the stack.

Key Points:

1. Push and Pop Operations:
    1.1. Push: Adds an element to the top of the stack.
    1.2. Pop: Removes the topmost element from the stack.

2. Top and IsEmpty Operations:
    2.1. Top: Retrieves the top element without removing it.
    2.2. IsEmpty: Checks if the stack is empty.

3. Memory Allocation:
    3.1. Stacks use contiguous memory and can be implemented using arrays or linked lists.

4. Applications of the Stack:
    4.1. Undo/Redo operations in text editors.
    4.2. Expression evaluation and conversion (infix to postfix/prefix).
    4.3. Backtracking problems like maze solving or pathfinding.
    4.4. Function call management in recursion.

5. Real-life Analogy:
    5.1. Like a stack of plates, where you add/remove plates from the top.

Q9. What is a binary search tree (BST) and how is it different from a binary tree? (5 marks)

Ans: A binary search tree (BST) is a special type of binary tree that maintains a strict ordering property where:

1) The left child contains values less than the parent node.
2) The right child contains values greater than the parent node.

    Key Points:

    1. Node Arrangement: BST follows the property that allows faster searching, insertion, and deletion operations.
    2. Binary Tree vs BST: A binary tree can have any random arrangement of nodes, while a BST follows a specific order.

3. Time Complexity: BST operations such as search, insertion, and deletion take O(log n) in an average-case scenario.
4. Degenerate BST: In the worst case, a BST may become skewed and behave like a linked list with O(n) complexity.
5. Application of BST: Used in implementing maps, sets, and maintaining sorted data.

Q10. Define dynamic programming and give an example. (5 marks)
Ans: Dynamic programming (DP) is a technique used to solve problems by breaking them down into overlapping sub-problems and storing the results to avoid redundant calculations.

Key Points:

1. Memoization vs Tabulation:
   1.1. Memoization: Top-down approach storing results of sub-problems.
   1.2. Tabulation: Bottom-up approach filling the table iteratively.

2. Optimal Substructure:
   2.1. DP is applicable when a problem can be solved by combining solutions to smaller sub-problems.

3. Overlapping Subproblems:
   3.1. Recurrence relations lead to recomputation, which DP avoids by storing intermediate results.

4. Example—Fibonacci Sequence:
   4.1.1. Recurrence relation: $F(n)=F(n-1)+F(n-2)F(n) = F(n-1) + F(n-2)$
   $F(n)=F(n-1)+F(n-2)$
   4.1.2. Using DP, the time complexity reduces from O(2^n) to O(n).

5. Applications of DP:
   5.1. Knapsack problem, longest common subsequence, and matrix chain multiplication.

Q11. What is a graph? Explain BFS and DFS traversal techniques. (5 marks)
Ans: A graph is a data structure consisting of nodes (vertices) and edges that connect pairs of nodes. Graphs can be:

1) Directed: Edges have direction.
2) Undirected: Edges have no direction.

Key Points:

1. Breadth-First Search (BFS):
   1.1. Explores all neighbors of a node before moving to the next level.
   1.2. Implemented using a queue.
   1.3. Time Complexity: O(V + E)
2. Depth-First Search (DFS):

2.1. Explores as far as possible along a branch before backtracking.
2.2. Implemented using recursion or a stack.
2.3. Time Complexity: O(V + E)

3. Applications of BFS:
3.1. Shortest path finding in an unweighted graph.
3.2. Network broadcasting and social media suggestions.

4. Applications of DFS:
4.1. Cycle detection, topological sorting, and solving puzzles.

5. Graph Representation:
5.1. Adjacency list (space-efficient) and adjacency matrix (faster lookups).

Q12. What are heap data structures and their types? (5 marks)

Ans: A heap is a complete binary tree used to implement priority queues where the parent node satisfies a specific ordering property.
Key Points:

1. Types of Heaps:
1.1. Max-Heap: The Parent node is always greater than its children.
1.2. Min-Heap: The Parent node is always smaller than its children.

2. Heap Property:
2.1. In a max-heap, the largest element is at the root, while in a min-heap, the smallest element is at the root.

3. Heap Operations:
3.1. Insertion: Adds an element and reorders the heap.
3.2. Deletion: Removes the root and reorganizes the heap.

4. Applications of Heaps:
4.1. Dijkstra's algorithm for shortest path.
4.2. Huffman coding for data compression.
4.3. Task scheduling in operating systems.

5. Time Complexity:
5.1. Insert and delete operations take O(log n), while access to the root takes O(1).

Q13. What is hashing and explain its applications? (5 marks)

Ans: Hashing is a technique used to map data of arbitrary size to a fixed-size value using a hash function. It provides efficient insertion, deletion, and search operations.
Key Points:

1. Hash Table Structure:
   1.1 Stores key-value pairs where a hash function maps the key to an index.

2. Collision Handling Techniques:
   2.1 Chaining: Uses linked lists to handle multiple keys mapping to the same index.
   2.2 Open Addressing: Finds an alternative index when a collision occurs.

3. Hashing Functions:
   3.1 Converts input data into a fixed-length hash code.

4. Applications of Hashing:
   4.1 Implementing dictionaries, symbol tables, and database indexing.

5. Time Complexity:
   5.1 Average case: O(1) for insertion, deletion, and search.
   5.2 Worst case: O(n) when collisions are high.