Important Questions for First Internal Exam

Unit-I

1. Define the terms "data," "entity," and "information" in the context of data structures.
2. Define abstract data type (ADT) and explain its significance in data structures. Provide an example of an ADT and its associated operations.
3. What is a data structure? Discuss the importance of data structures in programming and distinguish between linear and non-linear data structures. Provide examples of each type and explain their characteristics.
4. Define an algorithm and differentiate it from a program. Discuss the key properties of an algorithm. Explain the algorithm design techniques such as divide and conquer, dynamic programming, greedy algorithms and backtracking. Provide examples to illustrate each technique.
5. Define an array and discuss its applications in programming. Explain the concepts of single-dimensional and multidimensional arrays. Discuss the representation of arrays in row major order and column major order. Also discuss the address formulae for 1-D and 2-D arrays.
6. Explain the concept of sparse matrices and their representation. Discuss how sparse matrices can be stored efficiently in memory.

7. Compare and contrast the array implementation and pointer implementation of singly linked lists. Explain the operations (insertion, deletion, traversal) on a linked list.
8. Discuss the concept of a doubly linked list. Explain its advantages over a singly linked list and describe the operations that can be performed on a doubly linked list.
9. Explain the concept of a circularly linked list. Discuss its characteristics and the benefits it offers compared to linear linked lists.
10. Describe the operations (insertion, deletion, traversal) that can be performed on linked lists.
11. Explain the polynomial representation of a linked list. Discuss how addition, subtraction, and multiplication of single-variable polynomials can be performed using linked lists.

<u>**Unit - II**</u>

Stacks:

1. Define the abstract data type (ADT) "stack" and explain its characteristics. Discuss the importance of stack data structure in programming.
2. Explain the primitive stack operations: push, pop and peek using array and Linked list.
3. Describe the algorithms of converting an infix expression to postfix and prefix expressions using an example.

4. With the help of an example, explain the step-by-step process of evaluating a postfix expression using a stack.
5. What is recursion? Discuss the different types of recursion. Explain the concept of tail recursion and its significance in recursive algorithms. Provide an example to illustrate the use of tail recursion.
6. Explain the problem of the Towers of Hanoi. Discuss the recursive algorithm's steps and complexity.

Queues:

1. What is a Queue? Describe the operations of insertion, deletion and display in a queue checking if the queue is full, and checking if the queue is empty.
2. Explain the concept of circular queues and their advantages over linear queues. Discuss the implementation of circular queues using arrays and linked list.
3. What is a Double-ended queue? Discuss enqueue and dequeue operations in a double ended queue.
4. Explain the concept of a priority queue and its applications. Discuss the implementation of a priority queue using arrays or linked list.

Searching:

1. Define the concept of searching and its importance in data structures. Discuss sequential search and its implementation.
2. Explain Binary search using a suitable example. Provide step-by-step explanation for each step and also discuss its time complexity.

Sorting:

1. Explain the difference between the following:
    1. Internal and External Sorting
    2. In place and out-of-place sorting
    3. Stable and Unstable sorting
    4. Adaptive and Non-adaptive sorting
2. Explain the working of following sorting algorithms with suitable examples and also discuss their time complexity:
    1. Bubble Sort
    2. Selection Sort
    3. Insertion Sort
3. Explain the following linear time sorting algorithms: counting sort, radix sort and bucket sort. Discuss their steps, time complexity, and provide examples to illustrate their working.
4. Discuss the divide and conquer approach to problem-solving. Explain the step by step working of merge sort and quick sort algorithms with suitable examples and discuss their time complexities.