|  |  |
| --- | --- |
| **SESSION** | **FEB-MARCH 2025** |
| **PROGRAM** | **BACHELORS OF COMPUTER APPLICATIONS (BCA)** |
| **SEMESTER** | **II** |
| **COURSE CODE & NAME** | **DCA 1207 DATA STRUCTURES** |
|  |  |
|  |  |

**Set-I**

**Q1. What do you understand by Algorithm Complexity? Discuss Time and Space Complexity in detail by taking suitable examples**

**Ans 1.**

**Algorithm Complexity**

Algorithm complexity refers to the measure of the amount of computational resources that an algorithm uses during its execution. These resources are primarily time and space. Complexity analysis helps us to predict how an algorithm will perform when the size of the input increases. It is essential to compare algorithms and choose the most efficient one for a given problem.

**Time Complexity and Its Significance**

Time complexity refers to the amount of time an algorithm takes to complete as a function of the length of the input. It helps in evaluating the performance of an algorithm in terms of speed.

Its Half solved only

Buy Complete assignment from us

**Price – 190/ assignment**

**MUJ Manipal University Complete SolvedAssignments MARCH 2025**

buy cheap assignment help online from us easily

we are here to help you with the best and cheap help

**Contact No – 8791514139 (WhatsApp)**

**OR**

**Mail us-** [bestassignment247@gmail.com](mailto:bestassignment247@gmail.com)

**Our website -** [www.assignmentsupport.in](http://www.assignmentsupport.in)

**Q2. Write an algorithm to find a particular number in an array and replace it with some other value.**

**Ans 2.**

**Algorithm to Find and Replace an Element in an Array**

**Understanding the Problem**

In many real-world applications, we often need to search for a particular element in an array and update or replace it with a new value. This is a basic but important operation in array manipulation. The process involves two key steps: searching and replacement. We must scan the array, identify the element to be replaced, and then substitute it with the new value.

**Steps of the Algorithm**

To solve this problem, we can use a simple linear search technique. The array is scanned from

**Q3. Explain the working of a Queue data structure. What are its applications in real-world scenarios?**

**Ans 3.**

**Introduction to Queue Data Structure**

A Queue is a linear data structure that follows the FIFO (First-In, First-Out) principle. In a queue, the element inserted first is the one to be removed first. It is similar to a real-world queue, such as a line at a ticket counter, where the first person in the line is served first. A queue supports two primary operations: enqueue, which adds an element at the rear, and dequeue,

**Set-II**

**Q4. What is a linked list and its types? Discuss the benefits of using them over array in detail.**

**Ans 4.**

**Linked List and Its Types with Advantages Over Arrays**

**Linked List**

A Linked List is a dynamic data structure used to store a collection of elements, where each element, called a node, contains two parts: data and a pointer (or reference) to the next node in the sequence. Unlike arrays, linked lists do not store elements in contiguous memory locations. This dynamic behavior allows flexible memory usage, making linked lists suitable for

**Q5. What is a doubly circular queue? Write an algorithm to display the contents of the circular queue.**

**Ans 5.**

**Doubly Circular Queue and Algorithm to Display Contents**

**Doubly Circular Queue**

A doubly circular queue is a special type of queue that combines features of both a doubly linked list and a circular queue. In this structure, each node contains three fields: data, a pointer to the next node, and a pointer to the previous node. Unlike a regular doubly linked list, the last node’s next pointer points to the first node, and the first node’s previous pointer points to the last node, forming a closed circular loop. This allows traversal in both forward and backward directions without reaching a null pointer, making it highly efficient for applications requiring constant

**Q6. Write an algorithm for Merge Sort and explain its divide-and-conquer approach. 10**

**Ans 6.**

**Merge Sort Algorithm and Its Divide-and-Conquer Approach**

**Merge Sort**

Merge Sort is a popular and efficient sorting algorithm that follows the divide-and-conquer paradigm. It works by dividing the input array into smaller sub-arrays, sorting each sub-array recursively, and then merging the sorted sub-arrays to produce the final sorted array. It is known for its stable sorting and predictable time complexity of O(n log n), which makes it suitable for large datasets and complex systems.

**Divide-and-Conquer Approach in Merge Sort**

The merge sort algorithm works by breaking down a complex problem into smaller, manageable