Array

Assignment 1:

1. Write a program to find the maximum element in an array.

2. Implement a function to reverse an array in-place.

3. Create a program to find the intersection of two arrays.

4. Write an algorithm to rotate an array by a given number of positions.

5. Implement an algorithm to find the missing number in an array of integers from 1 to N.

Array

Assignment 2

6. *Write a function to remove duplicates from a sorted array.*

7. *Implement an algorithm to find the majority element in an array.*

8. *Create a program to find the longest subarray with a sum less than or equal to a given value.*

9. *Write a function to find the contiguous subarray with the largest sum (Kadane's algorithm).*

10. *Implement an algorithm to search for an element in a sorted and rotated array (rotated binary search).*

Assignment 3

11. Create a singly linked list and write a function to reverse it.

12. Implement a program to detect if a linked list has a cycle.

13. Write a function to merge two sorted linked lists into a single sorted list.

14. Implement an algorithm to find the nth node from the end of a linked list.

15. Create a program to delete a node with a given value from a linked list.

16. Write a function to check if two linked lists intersect, and if they do, find the intersection node.

17. Implement a function to add two numbers represented by linked lists (e.g., 342 + 465 = 807).

Assignment - 4

18. Implement the merge sort algorithm.

19. Write a program to perform the quicksort algorithm.

20. Implement the heapsort algorithm.

21. Create a function to sort an array using bubble sort.

22. Implement the radix sort algorithm for integers.

Assignment 5

23. Implement a binary search tree and write functions for insertion and deletion.

24. Write a program to find the height of a binary tree.

25. Implement an algorithm to check if a binary tree is a binary search tree (BST).

26. Create a function to find the lowest common ancestor (LCA) of two nodes in a binary tree.

Assignment 6

27. Write an algorithm to perform an in-order traversal of a binary tree without recursion.

28. Implement a function to find the kth smallest element in a BST.

29. Create a program to serialize and deserialize a binary tree.

30. Implement an AVL tree and perform insertions and rotations.

Assignment 7

39. Create a simple hash table with basic operations (insert, delete, search).

40. Implement a hash map with collision resolution (e.g., chaining or open addressing).

IT-361 (Data Structures Laboratory)

Experiment-1: Read data from FILE into an array. Write an implementation of bubble sort that can be used to sort the array. Use library function qsort to sort the array on i) rollNumber ii) name and iii) total of marks and save them in separate output files. Compare performance between your bubble sort implementation and qsort.

Experiment-2: Implement an AVL tree using pointers. Read data from File into your AVL tree, generating a "verbose" of steps during inserting each student data in the tree. Generate output files for pre-, in- and post-order traversal of your tree.

Experiment-3: Repeat Experiment-2 but with a Heap implemented using an array.

Experiment-4: Demonstrate the efficacy of a stack in solving a) postfix expression evaluation and b) Stock Span problems. Prepare you own input files to prove the correctness of your programs. There should be output files tracing the different parts of your implementation at different debug levels.

Experiment-5: Demonstrate the efficacy of Prim's and Kruskal's minimal spanning tree algorithms. Prepare you own input files to prove the correctness of your programs. There should be output files tracing the different parts of your implementation at different debug levels.

Experiment-6: Demonstrate the efficacy of Dijkstra's minimum path algorithm. Prepare you own input files to prove the correctness of your programs. There should be output files tracing the different parts of your implementation at different debug levels.