1. Find the number of occurrences of a given integer NUM from a pre-defined array.
2. Reverse the array and store it in a new array REV
3. Write a user-friendly program to create an array of size N, & perform the following operations
    a. Insert
    b. Delete
    c. Display
4. Find the largest and the second largest number from an array of integers without sorting the array.
5. Write a program to find the factorial of a number using recursion.
6. Accept an integer NUM from the user and display its address and value using the pointers.
7. Implement a pointer to an array to display all the array elements alongside the addresses.
8. Accept an array of integer ARR[ ] from the user to implement a call by reference and find the sum of the odd numbers and even numbers separately.
9. Design a 2D matrix of size 4x4 and display the elements using a pointer to the array.
10. Write a program to reverse a string without using any in-built functions.
11. Write a program to check whether a given string is in palindrome order or not without using in-built functions.
12. Write a program of your choice to properly demonstrate the usage and differences of Local and Global variables.
13. Demonstrate the usage & implementation of a *struct*. Accept integer, float,

and character data and display them.

14. Implement a structure to accept the data of an employee mentioned below and display them in a properly formatted manner:
    a. empId
    b. empName
    c. basic
    d. da: 25% of basic
    e. hra: 15% of basic
    f. ta: 10% of basic
    g. gross : basic + da + hra + ta

15. Implement *a* structure to accept the data of 3 students (*without using an array of structure/pointer to structure*) and display them in a properly formatted manner
    a. enrollmentNo
    b. regNo
    c. studName
    d. studSem
    e. studCGPA
16. Create a structure for a company that manufactures mobile phones. Create a repository of information that consists of the following data:
    a. prod_name
    b. prod_code
    c. prod_price
    d. prod_screensize
    e. prod_color

    Now, accept data for 5 different products and display them in a well-formatted manner. Use structure and arrays.
17. Implement *Dynamic Memory Allocation(DMA)* to demonstrate the usage and functionalities of the following:

a. malloc()

b. calloc()

c. free()

18. Implement *DMA* to accept N integer data and find the sum of the composite integers only. Also, display the memory allocations along with the data.

19. Implement *DMA* and create a memory location of size N. Accept the data by passing the base address of the allocation to a function and display them along with the memory locations inside the function.

20. Demonstrate the implementation of a Singly linked list. Create 4 nodes and display all the data using the first node only.

21. Consider a scenario of Singly Linked List and write down in your own words how you plan to design a user-friendly and menu-driven program for the same.

22. Write down in your own words:

    a. What is Validation?

    b. What is the requirement for validation in a program?

    c. How do you plan to handle the pointers properly in a Singly Linked List program?

23. Create a user-friendly and menu-driven singly linked list program to perform the following operations:

    a. Insert a node at the rear position

    b. Insert a node at the front position

    c. Insert a node at any position

    d. Delete a node from the rear position

    e. Delete a node from the front

    f. Delete a node from any position

    g. Traverse the list from both the beginning and end.

    h. Search any node based on position

    i. Search any node based on the value of the data of the node

    j. Display the elements

24. Implement (23) for a Singly Circular Linked List

25. Implement (23) for a Doubly Linked List

26. Implement a binary search tree & perform the following traversals

    a. Inorder
    b. Preorder
    c. Postorder

27. Implement the following search algorithms:
    a. Linear Search
    b. Binary Search
28. Implement the following sorting algorithms
    a. Bubble Sort
    b. Selection Sort
    c. Insertion Sort