DS TUT Assignments

1] Implement an array program for the following: Input: m* n matrix
a. Find saddle point in the Matrix.
B. Magic square Matrix. (Check)
c. Represent the given matrix in its Sparse form.

2] Use time.h to calculate the time taken by a program to run(take any program and compare different methods)

3] Write a program to perform the following operations :
a. Bubble sort - Sort and count swaps
Track intermediate states(pass 1 ka last array, pass 2 ka last array)
Custom order sorting
Optimize bubble sort

b. Insertion Sort
Tracking version
Binary search optimization
Insert element

c. Selection sort
Sort in descending order
Partial sorting
Min swap to sort
Sort by custom weights

4] Implement quick sort as well as randomized quick sort.
Extra: Handle duplications of the element (Modify the quick sort to handle an array with many duplicate elements)

5] Write a program to implement Radix Sort and Shell Sort

6] Write a program for the following operations :
a. Reverse a linked list iteratively and recursively (in a given order)
b. Detect and remove a loop in a linked list
c. Find the middle of the linked list in one pass
d. Merge two sorted linked lists for a given order

e. Check if a linked list is a palindrome.

7] Apply a Single linked list on all types of sort

8] Implement a Polynomial addition and multiplication using Linked Lists.

9] Implement SLL, DLL, and CLL for the following operations, create insert, delete, display, and reverse for the following operations. (start end and random)

10] Write a program to implement a parenthesis checker using stack
Mathematical equation - ((((a+b)*c)+d)-e)

11] Write a program to convert the given Infix expression into its Equivalent Prefix and Postfix form using Stack.

12] Write a program to implement static stack and dynamic stack (using a linked list)

13] Write a program to demonstrate any application of priority queue

14] Write a program to implement to perform basic operations on tree data structure

15] WAP to create a Binary tree and perform non-recursive Preorder, Inorder, and Postorder traversal.