Current Lab Discussion

Array

An array is a fixed-size data structure that stores a collection of elements of the same data type in contiguous memory locations. Arrays can be accessed using an index, which is a non-negative integer that corresponds to the position of the element in the array. Arrays are often used to store lists of data, such as the names of students in a class or the scores of a game.

Queue

A queue is a linear data structure that follows the First-In-First-Out (FIFO) principle. Elements are added to the queue at the back and removed from the queue at the front. Queues are often used to store tasks that need to be processed in a specific order, such as print jobs in a printer queue or customer orders in a restaurant.

Stack

A stack is a linear data structure that follows the Last-In-First-Out (LIFO) principle. Elements are added to the stack at the top and removed from the stack at the top. Stacks are often used to implement function calls in programming languages and to maintain a history of states in algorithms.

Linked List

A linked list is a dynamic data structure that stores a collection of elements in a linear sequence. Each element in a linked list is called a node, and each node contains a value and a reference to the next node in the sequence. Linked lists are often used to implement dynamic lists and queues, as well as more complex data structures such as trees and graphs.

Linear Search

Linear search is a simple search algorithm that works by sequentially searching through a list of elements until the target element is found or the end of the list is reached. Linear search is inefficient for large lists, but it is simple to implement and can be used for any type of data structure.

Binary Search

Binary search is a more efficient search algorithm that works by repeatedly dividing a sorted list in half until the target element is found or the list is empty. Binary search is only efficient for sorted lists, but it is much faster than linear search for large lists.

TASK-01

1st declare a 2D array and take input of rows and columns from a user. Then declare a 1D array and its size depends upon the size of 2D array's rows. Then save the sum of each row in a 1D array .

Code:

```
#include<iostream>
using namespace std;
int main()
{
    int row,column;
    cout<<"Enter number of rows:"<<endl;
    cin>>row;
    cout<<"Enter number of columns:"<<endl;
```

```
        cin>>column;
        int a[row][column];
        for(int i=0;i<row;i++)
        {
            for(int j=0;j<column;j++){
            cout<<"Enter row "<<i+1<<" & column "<<j+1<<endl;
            cin>>a[i][j];
            }

        }
        int sumofrow[row];
        for(int i=0;i<row;i++)
        {
            int sum=0;
            for(int j=0;j<column;j++){
            sum=sum+a[i][j];
            sumofrow[row]=sum;
            }
            cout<<"Sum of row "<<i+1<<" is: "<<sumofrow[row]<<endl;
        }
        return 0;
    }
```