Q1) what is the Data structure? Differentiate between linear and Non-linear Data structure.

→

i) Data structure can be defined as the group of data elements which provides an efficient way of storing and organising data in the computer so that it can be used efficiently.

ii) ex: arrays, linked list etc.

*	linear data structure	non-linear data structure
i)	Every item is related to its previous and next item	i) Every item is attached with many other items.
ii)	Data is arranged in linear seqvense	ii) Data is not arranged in seqvense
iii)	Data items can be traversed in a single run	iii) Data can not be traversed in a single run.
iv)	memory utilization is inefficient	iv) memory utilization is efficient
v)	It is single-level	v) It is multi level.
vi)	It is Easier to implement	vi) ~~not~~ It is difficult to implement.
vii)	Ex. Array, linked list queue, stack	vii) ex. tree graph.

Q3] write an algorithm to delete node in linked list

⇒ step 1: copy the address of first node
step 2: move the head to the second node of the linked list
step 3: Disconnect the connection of first node to second node.
step 4: free the memory occupied by the first node.

OR.

```
%/% input: head of the linked list.
Begin:
    if (head != NULL) then
      todelete ← head
      head ← head.next
      unalloc todelete)
    End if
End.
```

4] write an algorithm for linear search. Derive recurrence relation and find the time complexity.

⟹

step1 set i to 1
step2 if i > n then go to step 7
step3 if A[i] = x then go to step 6.
step4 set i to i+1
step5 go to step 2
step6 print Element x Found at index i and go to step 8
step7 print element not found
step8 Exit.

recurrence Relation

T(n) - Time complexity.

$$T(n) = 2T(n-1) + 3n.$$

$$= 2^2 T(n-2) + (2+1) \cdot 3n$$

$$= 2^3 T(n-3) + (2^2+2+1) \cdot 3n$$

$$= \sum_{i=0}^{n-1} 2^i \cdot 3n$$

$$= \frac{2^n - 1}{2-1} \cdot 3n$$

$$T(n) = O(n \cdot 2^n).$$

Page No.
Date

Q 5] Explain the Asymptotic notation Big Oh omega and theta with suitable example

→ i] Asymptotic Notation is used to describe the running time of an algorithm

ii] How much time an algorithm takes with a given input, n.

iii] There are three different notation big O, big theta (Θ), and big omega (Ω)

iv] big-Θ is used when the running time is the same for all cases

v] big-O for the worst case running time.

vi] big-Ω for the best case running time

a) Big O: example.

6] Explain generalized linked list with Example. Also write algorithm to create singly linked list.

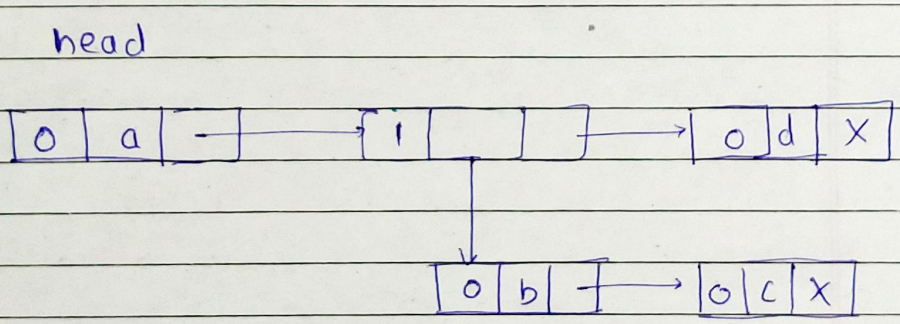
i] A generalized linked list L_i is defined as a finite sequence of $n >= 0$ elements

ii] $l_1, l_2, l_3, l_4, \ldots l_n$. such that l_i are either atom or the list of atoms

iii] Thus $L = (l_1, l_2, l_3, l_4, \ldots l_n)$
where n is total number of nodes in the list.

example. of GLL

(a, (b, c), d)



iv] when first field is 0, it indicates that the second field is variable.

v] If first field is 1 means the second field is a down pointer means some list is starting.

Page No.
Date

Algorithm

Step 1 start the program

Step 2 ~~start the program~~ get the choice from user.

Step 3 if the choice is to add records, get the data from the user and add them to the list

Step 4 if the choice is to delete records get the data to be deleted and delete it from the list.

Step 5 if the choice is to display number of records, count the items in the list and display.

Step 6. if the choice is to search for an item, get the item to be searched and respond yes if the item is found otherwise no.

Step 7 Terminate the program.

Page No.
Date

Q.8) write an algorithm to insert new node in linked list.

⇒

step 1 :- Declare a head pointer and make it as NULL

step 2 :- create a new node with the given data.

step 3 : make the new node points to the head node.

step 4 : Finally, make the new node as th head node.

Q.9. State and Explain characteristics of an Algorithm

⇒ It must have the following characteristics

i) clear and unambiguous :-
Algorithm should be clear and unambiguous. Each of its steps should be clear in all aspects and must lead to only one meaning.

ii) well-Defined inputs :
If an algorithm says to take inputs, it should be well-defined inputs.

iii) well-Defined outputs :
The algorithm must clearly define what output will be yielded and it should be well-defined as well

iv) Finite-ness :
The Algorithm must be finite it should not end up in an infinite loops or similar

v) Feasible :
The algorithm must be simple generic and practical, such that it can be executed upon will the available resources.

Page No.
Date

Q10 what is complexity analysis of an algorithm? Explain the notations used in the complexity analysis.

i) A technique to characterize the execution time of an algorithm independently from the machine, the language and the complier.

ii) It is useful for evaluating the variations of execution time with regard to the input data.

iii) It is ~~comparing~~ used to comparing algorithm

* The notation are used in the complexity analysis.

i) There are three different notations big O, big Theta (Θ), and big omega (Ω)

ii) big-O is used is used for the worst case running time.

iii) big theta (Θ) is used when the running time is the same for all cases

iv) big Ω for the best case running Tim

Q 11] Define.

1. ADT :-

ADT is a data structure and a set of operations which can be performed on it.

2] Data structure :-

The group of data elements which provides an efficient way of storing and organising data in the computer so that it can be used efficiently

3) Recursion :-

Recursion is a process in which a problem is defined in terms of itself.

4] Stack :-

A collection of items in which only the most recently added item may be removed. is called as stack.

5] linked list :-

a linear collection of data element whose order is not given by their physical placement in memory. is called as linked list.

Page No.
Date

Q12] write a pseudo c/c++ code to represent circular linked list as an ADT

=>

```
#include <iostream>
using namespace std;
struct Node {
{   int data;
    struct Node *Next;
};
  struct Node* head = NULL;
  void insert (int Newdata)
   {
      Newnode= New node();
      struct Node * ptr = head;
      Newnode -> data = Newdata;
      Newnode-> next = head;
   if (head != NULL)
  {
      while (ptr -> next != head)
        ptr = ptr -> next;
        ptr -> next = newnode;
      }
    else {
      newnode -> next = newnode;
       head = newnode;
  }
  void display ()
    {
      struct Node* ptr;
      ptr = head;
      do
      {  cout << ptr -> data << " ";
         ptr = ptr -> next;
      }
```

```
    while (ptr != head);
}
int main()
{
    insert(3);
    insert(1);
    insert(7);
    insert(2);
    insert(9);
    cout<<"The circular linked list is:";
    display();
    return 0;
}
```

output:

The circular linked list is: 9 2 7 1 3

Page No.
Date

Q13] write an algorithm to perform following operations on singly linked list.

step 1: Define three nodes one with the reference to the head node, and other two nodes as NULL

step 2: Now run a loop which will be used to traverse the linked list once until the next node does not become NULL

step 3: Now inside the loop, the first NULL node is defined as the next node to the head node

step 4: secondly, the next node of the head node is defined as the second NULL node.

step 5: Now, the second NULL node in step 4 is again redefined as the node which holds the reference to the head node in step 1

step 6: And finally, the node holding the reference to the head node in step 1 is made to hold the reference to the next node in the linked list and hence, the pointer now points to its previous node.

Page No.
Date

Q.14] write a Pseudo C/C++ code to concatenate two strings

⇒

```
#include <iostream>
#include <cstring>
using namespace std;

int main()
{
    char a[20], b[20];
    cout << "Enter string a: ";
    cin.getline(a, 20);

    cout << "Enter string b: ";
    cin.getline(b, 20);

    strcat(a, b)

    cout << "a = " << a << endl;
    cout << "b = " << b;

    return 0;
}
```

16] Define and explain the following terms

1) linear Data structures

i) A data structure is linear if all the elements are arranged in a linear order where the element are attached to its previous and next adjacent is called a linear data structure.

ii) In linear data structure, single level is involved.

iii) Therefore, we can traverse all the elements in single run only.

iv) linear data structures are easy to implement because computer memory is arranged in a linear way.

v) ex. array, stack, ques & linked list etc.

2) Non-linear data structure.

i) Data structures where data elements are not arranged seaventially or linearly are called non-linear data structure.

ii) In non linear data structure single level is not involved.

iii) Therefore, we can't traverse all the element in single run only.

iv) It is not easy to implement in comparisoon to linear data structure.

v) It utilizes computer memory efficiently in comparison to a linear data structure.

vi) example :- trees and graphs.

3) Time complexity :-

i) the amount of time taken by an algorithm to run, as a function of the length of the input. is called as time complexity.

ii) it measure the time taken to execute each statement of code in an algorithm

4) Space complexity.

i) The amount of memory used by the algorithm. to execute it completely and produce the result is called as space complexity.

Page No.
Date

8.16] Explain seaventhal search and binary search with appropriate example and compare their time complexity and space complexity.

1) Seaventhal search :-

i) The seaventhal search is also called as linear search.

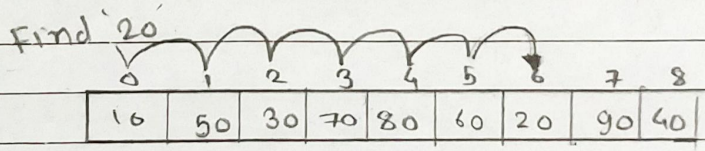
ii) A seaventhal search scans one item at a time, without jumping to any item

iii) start from the left most element of arr[] and one by one compare x with each element of arr[]

iv) if x matches with an element return the index.

v) if x doesn't match with any of elements. return

vi) example :-



linear search

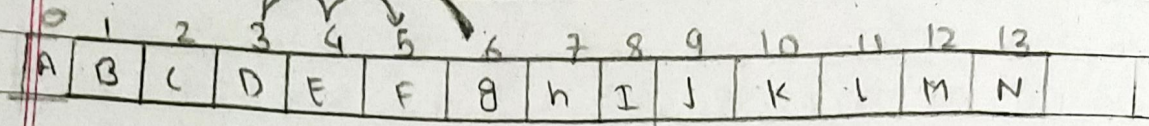
2] binary search :.

i) A binary search however, cut down your search to half as soon as you find middle of a sorted list

ii) The middle element is looked to check if it is greater than or less than the value to be searched.

iii) Accordingly, search is done to either half of the given list.

Find F



Binary search

- ~~Sequential search~~

Comparision

Algorithm	Time complexity	Space complexity
Seaquential search	O(n)	O(1)
Binary search.	O(log n)	O(1)

Page No.
Date

Explain two-dimensional arrays with row and column major implementation Explain address calculation in both cases with example.

i) The 2-D array is similar to that of 1-D array, but here we have two subscripts.

ii) The syntax of declaration of a 2-D array is

data-typ array-name[rowsize][columsize];

iii) Here rowsize specifies the number of and columsize represents the number of columns in array.

iv) The total number of array is rowsize * columnsize.

* Address calculation in Row

i) The following formula for calculating address of Row.

Address of A[I][J] = B + W * ((I - LR) * N + (J - LC)).

* Address calculation in column.

i) The following formula for calculating address of column.

Address of A[I][J] = B + W * ((J - LC) * M + (I - LR))

Example:

Q. Given an array [1.....10] [1.....15] with base value 100 and the size of each element is 1 Byte in memory

i] Find the address of arr[8][6] with the help of ~~column~~ row-major order.

ii] Find the address of arr[8][6] with the help of column-major order.

⇒ Given,

B = 100, W = 1, I = 8, J = 6, LR = 1

LC = 1

N = Number of column in matrix.

= upper bound - lower bound + 1

= 15 - 1 + 1

= 15

Address of row :

= B + W * (I - LR) * N + (J - LC)

= 100 + 1 * (8 - 1) * 15 + (6 - 1)

= 100 + 1 * 7 * 15 + 5

= 100 + 1 * 110

= 100 + 110

= 210.

Address of A[8][6] is 210.

column address :

= B + W * (J - LR) * N + (I - LC)

= 100 + 1 * (6 - 1) * 15 + (8 - 1)

= 100 + 1 * 5 * 15 + 7

= 100 + 1 * 57

= 157.

Page No.
Date

Q.18] write Pseudo C|C++ code to perform insert and delete operation on linear linked list

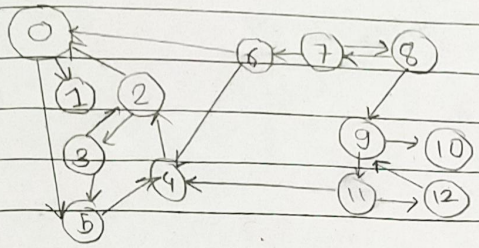
⇒

```
#include <iostream>
using namespace std;

class Node
{   int data;
    Node* Next;
};
  void print()
  {
     Node * ptr = Head;
  while (ptr != NULL)
    {
      cout << " Element : " << ptr→data << endl;
      ptr = ptr→next;
    }.
int main()
 {
     Node * Head;
      Node * second;
      Node * third;
      Node * NewNode;     — insertion

      Head = New Node();
      second = New Node();
      third = New Node();
      Newnode = New Node();

      Head → data = 5;
      Head → next = second;
```

```
second -> data = 6;
second -> Next = third;
third -> data = 7;
third -> next = Newnode
Newnode -> data = 8;
Newnode -> next = NULL;
  Print ();
  head = head -> Next      - Deletion.
  Print ();
  return 0;
}.
```

Page No.
Date

Q1) For the diagraph given below, obtain
1) Indegree and outdegree of each vertex
2) Its adjacency matrix
3) Adjacency list.



1) Indegree and outdegree of each vertex.

vertex	Indegree	outdegree.
0	2	3
1	1	0
2	2	2
3	1	2
4	3	1
5	2	1
6	1	2
7	1	2
8	1	2
9	2	2
10	1	0
11	1	2
12	1	1