Date

Data Structure

Classification Data Structure

↓ (Predefine)

Primitive Datatype

- → Integer
- → Real
- → Character
- → Boolean

(Derived from ba... data t...)

↓

Non-Primitive data type

- Linear DS
  - → Array
  - → Linked List
  - → Stack (LIFO)
  - → queue (FIFO)
    - Rear - Insert...
    - Front - Delete...
- Non Linear D...
  - → Tree
  - → Graphs

→ Complexity of linked list is $(n)^{Power}$

Abstract Data type is a Mathmatic model with the collection of operation define on that model.

Data structure deals with the Implements of various abstract data type for ex:

Stack, queue, ~~Linked~~ List.

Abstract Data Type

Goals of Data structure

1) Correctness
2) efficiency (space & Time)
3) Robustness (correct output)
4) Adoptability (able to adjust new cond^n)
5) Reusability &

Performance Analysis

1) Timing Complexity i.e. ex- OS / Networking
2) Space Complexity

* To provide satisfactory real time Responce (Time Complexity)
* Space for the Input also along With Algorithm / multiuser

Asymptotic Notations

To analays the complexity of algorithm in terms of time and space we can never provide an exact no. to define the time & space required by an algorithm instead we express it using some standard notations also known as Asymptotic Notations.

* endl / \n different

Page No.
Date

1) O big ohh ^ upper bound. (represent)

2) Ω omega lower bound

3) Θ ~~Theta~~ Theta. average bound.

* $1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < 2^n < 3^n - - - - n^n$

1) big Oh notation (big Oh O)

The fun. $F(n) = O(g(n))$ iff. and only iff ⇒
∃ (there exist). +ve Constant C & n_0 Such that $F(n) \le C * g(n)$ ∀ $n \ge n_0$

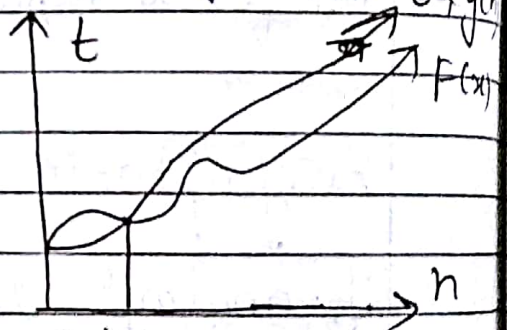
$f(x) = 2n+3$

$2n+3 \le ?$

$2n+3 \le 10n$

$F(x) \le c*g(n)$

$F(x) = O(n)$



$2n+3 \le 10n$

$5n^2$

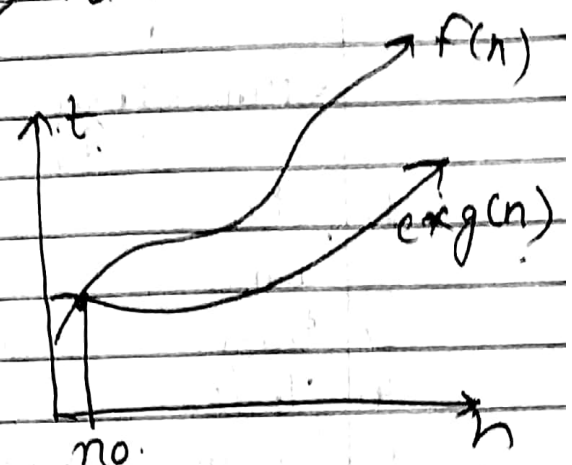
c g(n)

Page No. _____
Date _____

2) Omega notation Ω

The fun $f(n) = \Omega(g(n))$ iff and only if there exist some +ve constant c & n_0 such that
$f(n) \geq c * g(n) \quad \forall \; n \geq n_0$

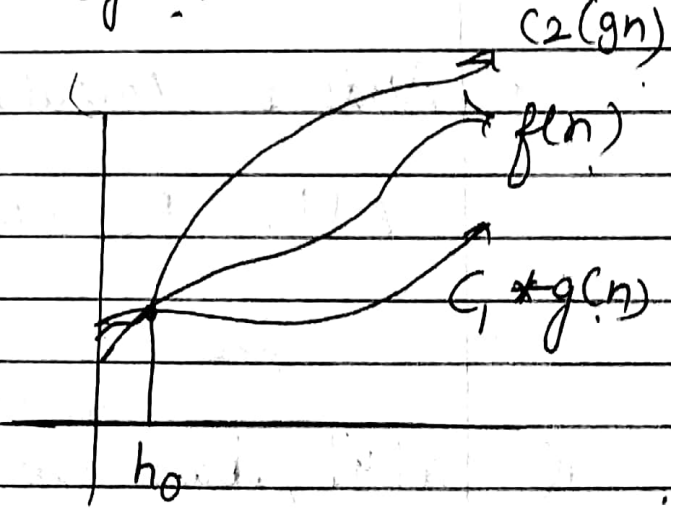
$f(x) = 2n+3$
$2n+3 \geq ?$
$2n+3 \geq sn \;/\; 4n/3n/2n/1n$
$f(n) = \Omega(n)$



3) Theta notation Θ

The fun $f(n) = \Theta(g(n))$ iff & only iff there exist some +ve constant c_1, c_2 & n_0 such that
$c_1 * g(n) \leq f(n) \leq c_2 g(n)$

* If Omega & Big oh is not equal than theta is not calculated



Array

An array is a finite, ordered and collection of homogeneous data element.

Terminology

1) Size :
2) type
3) base → address of the first element
4) Index
5) Range → lower bound, upper bound are called the boundaries of array.

In case of C A[100] LB=0 UB=99

In Case of Pascal - A: Array [-5 ... 19] of Integer

LB=-5 & UB=19

* Calculating the length of array :- UB-LB+1

* Address calculation in 1D Array

$A(K) = Base(A) + W(K-LB)$.

W = Size.

Q a[4], base=100, w=4, A[3]=?

A[3] = 100 + 4(3-0)

112

[a[n] = a[n-1]
if[a[i] = 25]

Page No.

Q Write a program to insert element at specific index of array

10 20 30 40 50 21 70 80 90 10 11

2-D Array

A[3][4]

Row Major

RM →

00	01	02	03
10	11	12	13
20	21	22	23

CM ↓

00	10	20	30
01	11	21	31
02	12	22	32

Column Major

* Row Major order

$A[K_1, K_2] = Base(A) + W(E_1 L_2 + E_2)$

Column Major order

$A[K_1, K_2] = Base(A) + W(E_2 L_1 + E_1)$

LB1 & UB1 are lower bound & upper bound of A1
Similarly LB2 & UB2 are lower & upper bound of A2
L1 & L2 be length of First and 2nd subscript

$L_1 = UB_1 - LB_1 + 1$

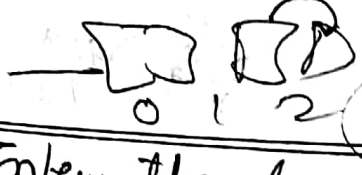
$L_2 = UB_2 - LB_2 + 1$

E_1 for the subscript K_1 the effective index E_1 of L_1 is computed as. $E_1 = K_1 - LB_1$

Page No.
Date

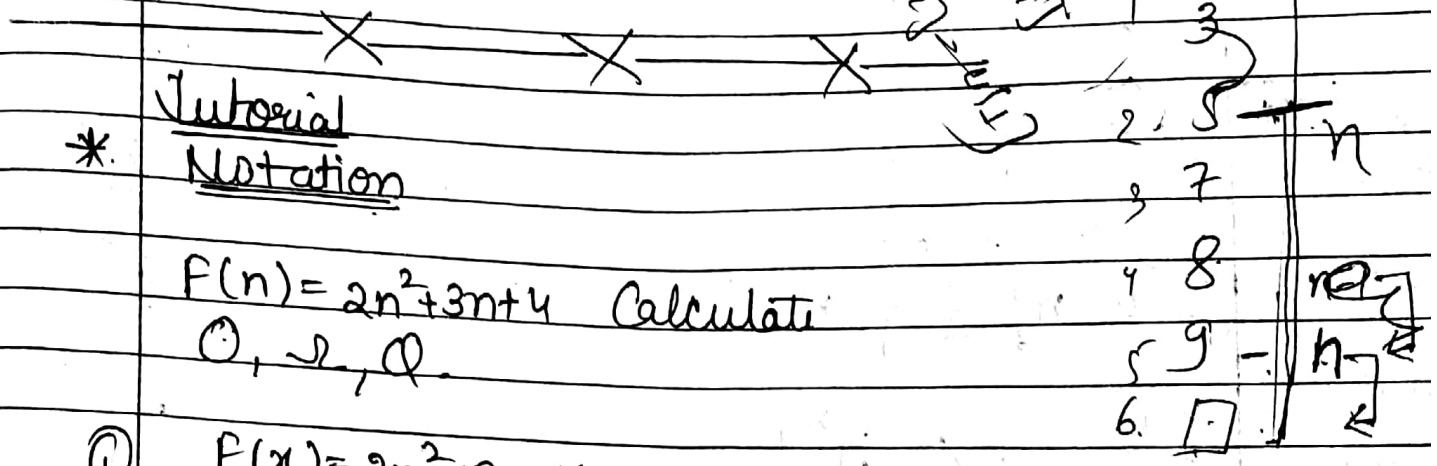
Effective index of E_2 is calculated as $K_2 - LB_2 = E$

Q $AX = A[4][3]$
Base



Page No. _____
Date _____

Enter the element of an array pls enter an element.
else array element befor inserting elem
enter the position where u want to insert
enter the element to insert
array list after inserting an element



Tutorial

* **Notation**

$F(n) = 2n^2+3n+4$ Calculate O, Ω, Θ.

① $F(x) = 2n^2+3n+4$

$F(n) \leq c * g(n)$.

$2n^2+3n+4 \leq 2n^2+3n^2+4n^2$

$9n^2$

$c \cdot g(n)$

$O(n^2)$

② Ohmega

$F(n) \geq c * g(n)$

$2n^2+3n+4 \geq 1 \cdot n^2$

$c * c(g)$

$\Omega(n^2)$

(iii) Theta Θ

$\Omega(n^2) \leq 2n^2+3n+4 \leq O(n^2)$

$\therefore \Omega(n^2) \leq \Theta(n^2) \leq O(n^2)$

Q $f(n) = n!$

(i) O big oh

$f(n) \leq c * g(n)$

$n! \leq n^n$

$O(n^n)$

(ii) ohmega. Ω

$f(n) \geq c * g(n)$

$n! \geq 1$

$\Omega(1)$

Q $f(n) = \log n \leq \log(1*1*1*1)$

$\leq \log(1 \times 2 \times 3 \cdots n)$

$\leq (\log n \times n \times n)$

$\log n^3$

$O(n \log n)$

(i) $O($

(ii) $\Omega(\quad)$

Ex:- 2D Array

A [4] [3]

B = 52722

Word = 2 Bytes

Cal. A [2] [1] = ?

Row major

7 x 2
14 + 52722

R [ ] = Base + w (E1 L2 + E2) = 52736

52722 + 2 (4 x 1 + 2)

52722 + 4 (6)

24.

2 x 3 + 91
7

52722
24
52746

By using formula

A [K1] [K2] = Base (A) + w (E1 L2 + E2)

A [2] [1] = 52722 + 2 (2 x 3 + 1)

= 52722 + 14

= 52736

L1 = 3 - 0 + 1 = 4

L2 = 2 - 0 + 1 = 3

E1 = 2 - 0 = 2

E2 = 1 - 0 = 1

By using colun major

A [2] [1] = 52722 + 2 (1 x 4 + 2)

= 52722 + 12

52734

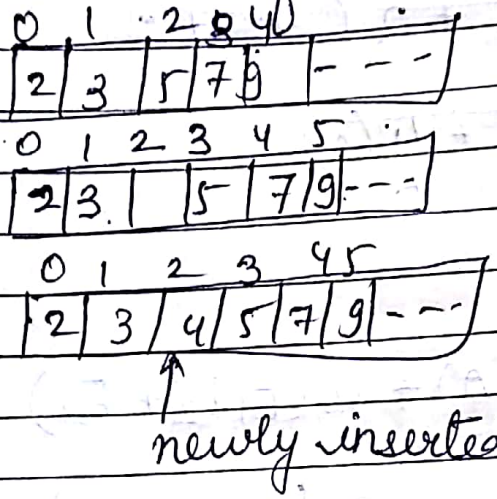
00	10	20
01	11	21
02	12	22
03	13	23

Page No.
Date

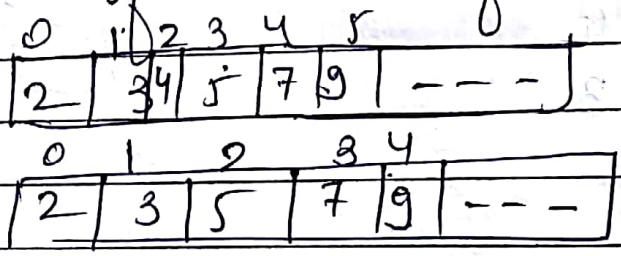
Operations of Array

Insertion
Deletion
Sorting
Searching
Merging
Traversing

★ Insertion of an element in array



★ Deletion of an element from array



Traversing - accessing elements of array atleast once operations of traversing - transpose, add Ds

Date

Linked list

Drawback of array

1) Exatra size of array which is not come in use than it is wastage of memory
2) Insertion and deletion is tredious task in array.
3) We define the size intionaly

Node

Data	Link

* We can elementate the drawback of array. Linked list support Dynamic memory allocation by creating Node at Run time.

→ Malloc
→ ~~Cattaloc~~ Calloc
→ Free~~dlloc~~
→ Realloc

(i) | 10 | Null | → single Node

(ii)

20	200	→	30	350	→	45	Null

200 350

* Linked list Refers to a linear Collection of data items. Storing a linked list in memory is to have two parts or
1) To store data call info field.
2) To store the address of next node called as link (Pointer field).

To add two node using Linked List.

Page No.
Date

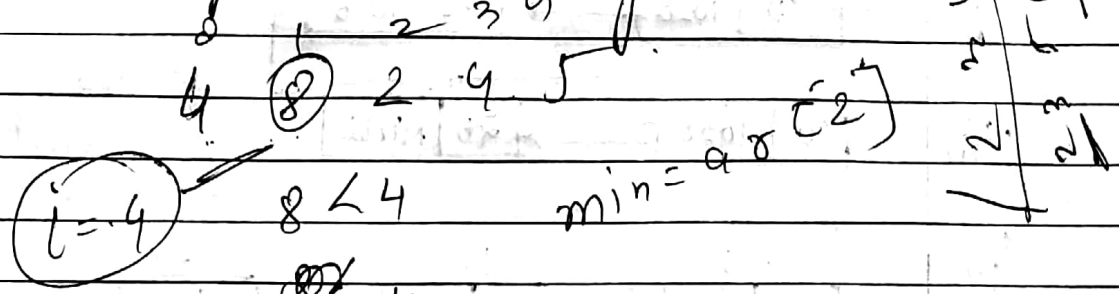
```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#include<Process.h>

Struct Node
{
    int info;
    Struct Node *link;
} node;

Void main()
{
    Struct node *node1, *node2;
    int data;
    node1 = (Struct node *) malloc (size of (n
node2 = (struct node*) malloc (size of (node));
printf ("Enter data for node1");
Scanf ("%d", &data);
node1 → info=data;
printf ("Enter data for node2");
Scanf ("%d", &data);
node2 → info=data;
node1 → link = node2;
node2 → link = null;
printf ("Address of Node1 is %u", node1);
printf ("Info of Node1 contains %d,
                    node→info);

printf ("Address store in link field of node1
        is %u", node1 → link);
printf ("Repeat the for node2)
```

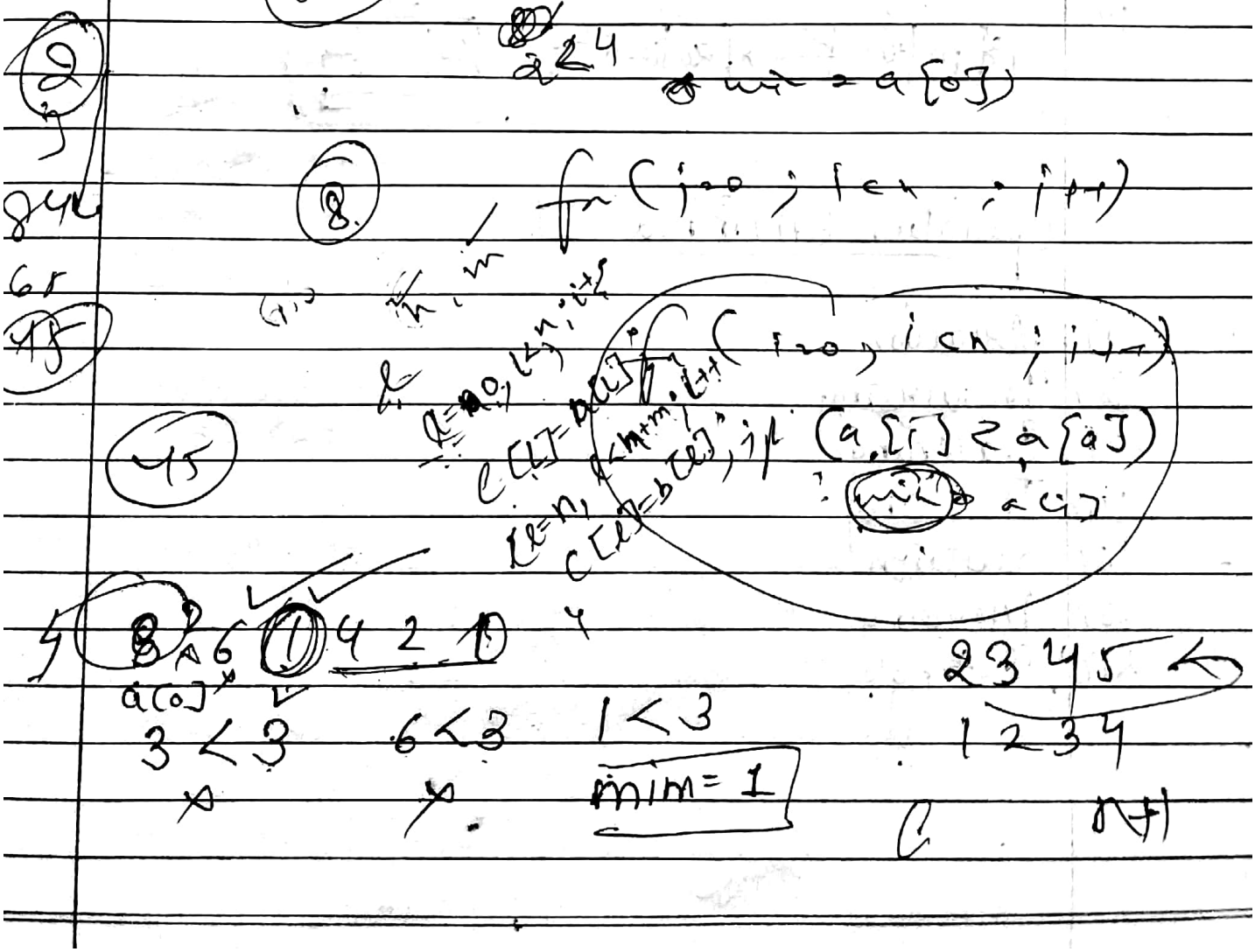
Page No. _____
Date _____

Q Consider an array of size n. write a program to find max and min no. from list of element.

Q Consider an array of size n :-
1) Insert element into array
2) Display
3) Search
4) Sort array element assending & decesending

Q Consider 2 arrays of size n m resp. write a program to merge these array.



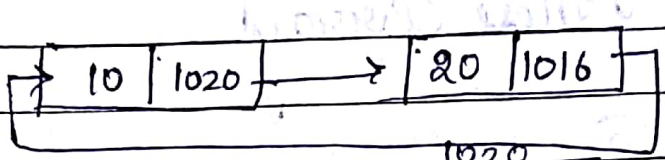


Types of Linked List
(Linear)

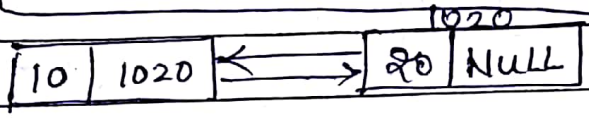
1) Single Linked List
2) Circular Linked List
Linear
3) doubly linear linked list
4) Circular doubly linear linked list

(i)

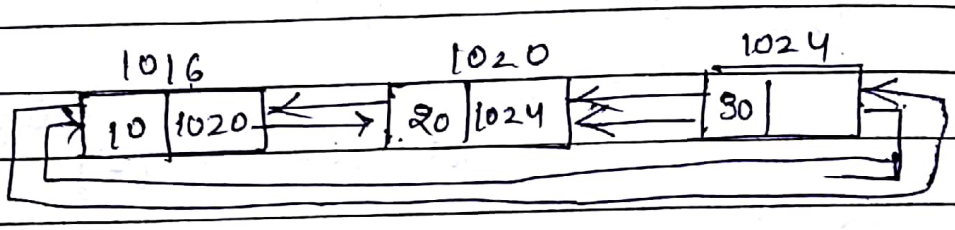
(ii)



(iii)



(iv)



Operation on Linked List

(1) Creation
(2) Traversing
(3) Searching
(4) Inserting
(5) deletion
(6) Merging
(7) Sorting

Page No. _____
Date _____

```
#Include<Stdio.h>
#Include<conio.h>
#include<alloc.h>

Struct node
{
  int info;
  Struct node *link;
};
struct node *first = NULL,
            *last = NULL;

Void create ();
Void display ();

Void main ()
{ int choice;
  Clrscr ();
  printf ("Enter choice");
  Scanf ("%d", &choice);
  printf ("Press 1. Create,
                2. display,
                3. Quit");
While (choice)
{
  Switch (choice)
    { case 1 : create (); break;
      Case 2 : display (); break;
      case 3 : exit ();
    }
```

```
printf ("Enter the choice");
printf ("1-create
         2-display
         3-Quit");
Scanf ("%d", &choice);
{

Void create ()
{
  Struct node *new = malloc sizeof (Struct n
printf ("Enter item to be inserted");
Scanf ("%d", &new -> info);
new -> link = null;

if (first == NULL)
    first = new;
else
    last -> link = new;
    last = new;
}
Void display ()
{
If (first == null)
  printf ("no element");
else
{ Struct node *temp = first;
  printf ("Element are");
  While (temp != NULL)
  { printf ("%d", temp -> info")
      temp = temp -> link;
  }
  Printf ("Null")
}}
```

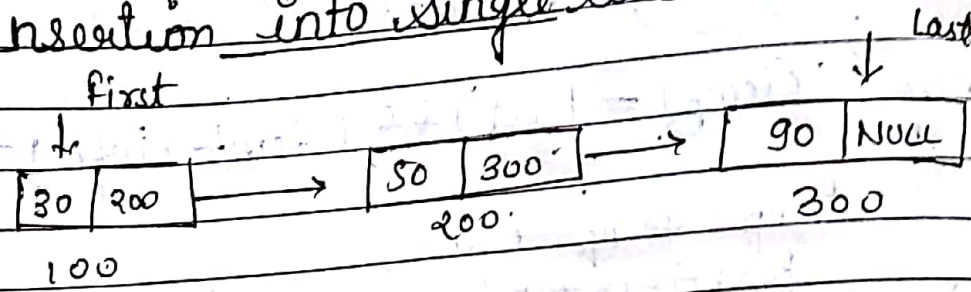
Searching in Single linklist

```
Void Search()
{
 if (First == Null)
 printf ("There were no element in the list");
 else
 { Struct node * temp. = First;
    int element, position = 1;
    printf ("Enter the element");
    scanf ("%d", &element);

 While ((temp != Last) && (temp → info != element))
    {
        temp = temp → link;
        position ++;
    }

 if (temp → info == element)
    printf ("the element is present in the position %d",
                position);
 else
    printf ("Element Not Found");
 }
}
```

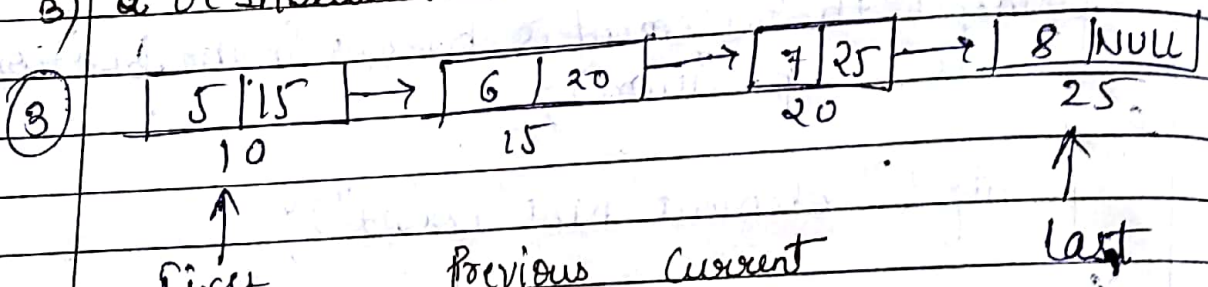
Traversing of Single Linear Linklist

```
while ((temp) = last) && (temp -> info) = element)
{
    temp = temp -> link;
    position ++;
}
```

Insertion into single linear linklist



1) at begining
2) at the end
3) a or Inbetween (location)



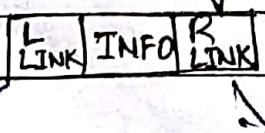
Previous Current

78 | NULL

Previous -> link = New

New -> link = Current

Insertion Doubly linklist



Left Link

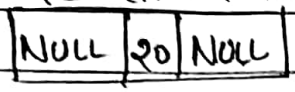
Right link

Page No. _____
Date _____

```
struct dnode
{
    int info;
    struct dnode * llink;
    struct dnode * rlink;
};
```

FIRST Node

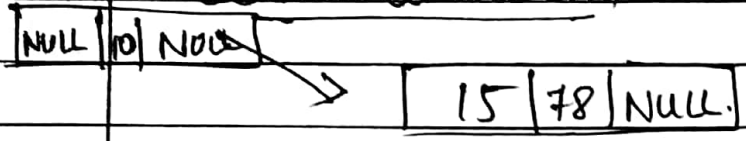
Insertion in First



New 25

New → rlink = FIRST.
FIRST → llink = New.
FIRST = New.

Insertion at the End



New → Llink = FIRST.
FIRST → Rlink = new
Last = New.

FIRST Node - Llink = NULL
Last Node - Rlink = NULL