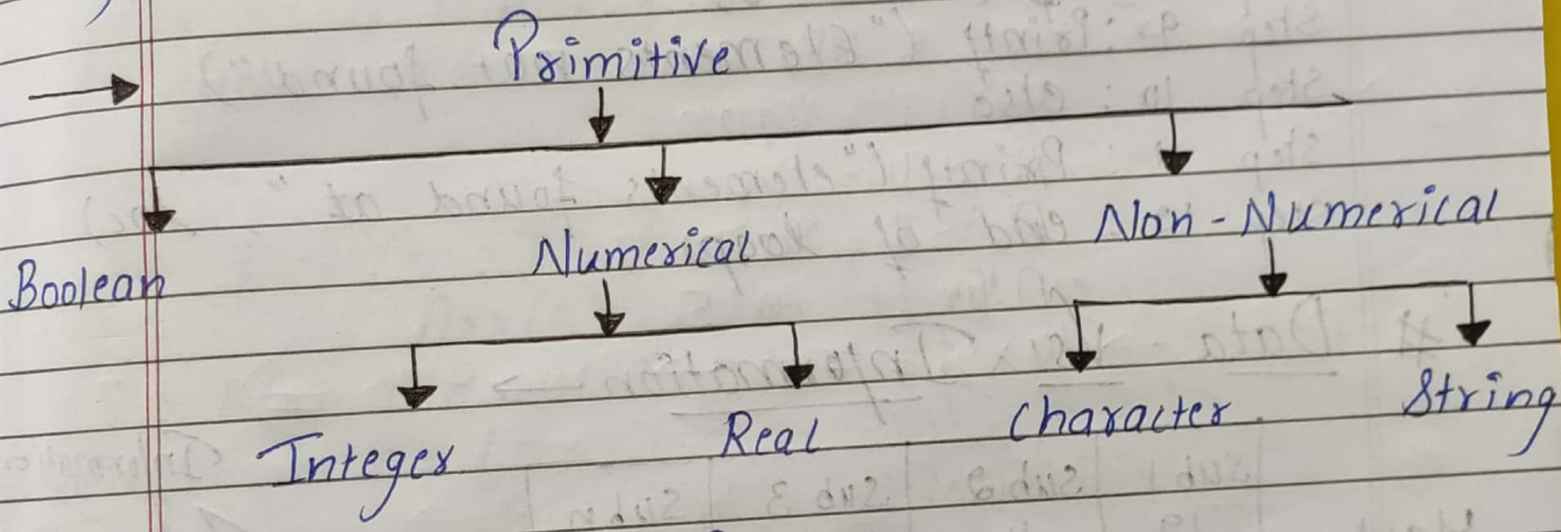
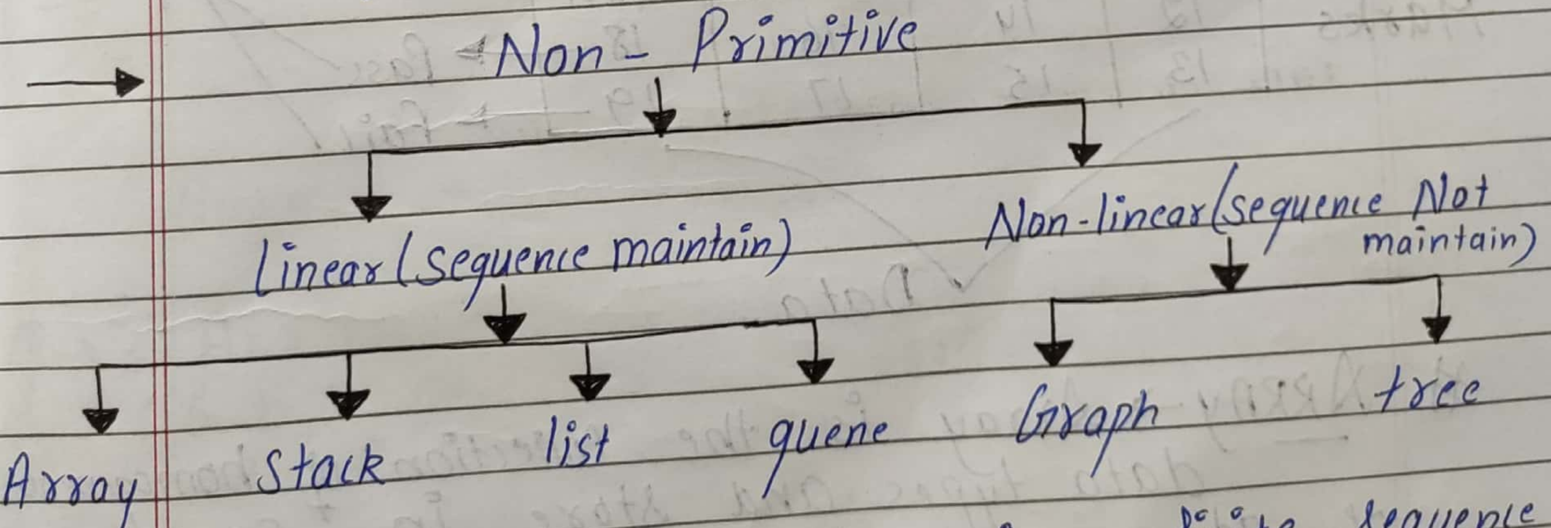
Data structure = Data + operation
(Insertion, deletion, increment, searching, decrement, sorting, comparison, merging etc.)

TYPES OF DATA →

There are two types of data →

1.) Primitive   2.) Non-Primitive

→ 

→ 
Non-Primitive
linear (sequence maintain)
Non-linear (sequence Not maintain)
Array
Stack
list
quene
Graph
tree

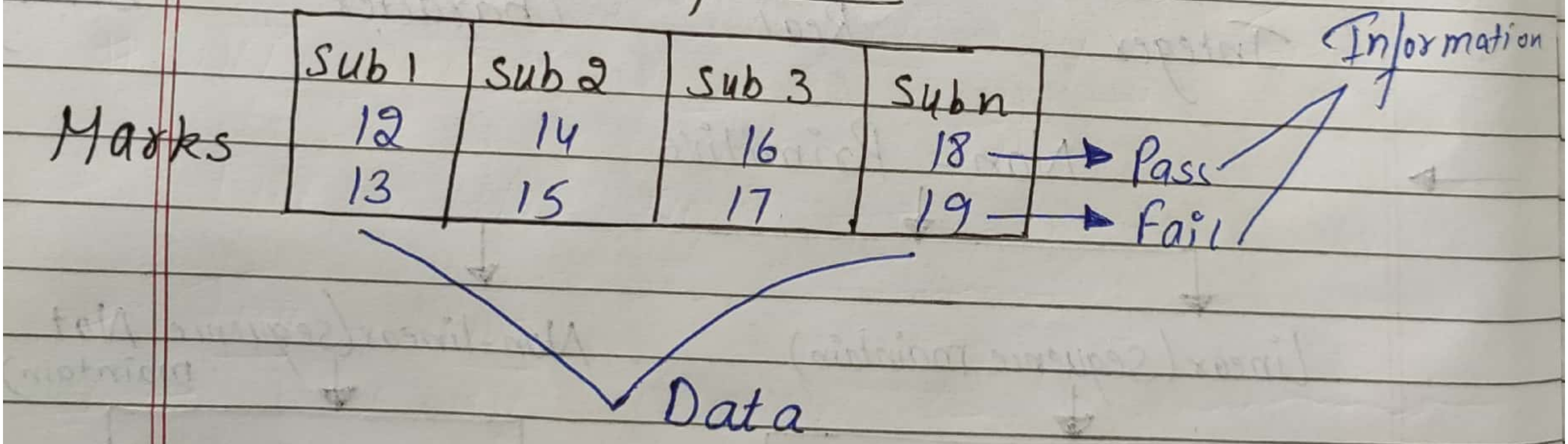
ALGORITHM → An algorithm is a finite sequence of instruction, each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time.

Scanned by CamScanner

Q → Write an algo to find a no. 'num' in the list of numbers sort in an array 'arr'.

Sol^n →
```
Step 1 : Start
Step 2 : loc = 0
Step 3 : loop , i = 1 to n step by 1   (incr by 1)
Step 4 : if (arr [i] = = num)
Step 5 :     loc = i
Step 6 :     break
Step 7 :
Step 8 : if (loc = = 0)
Step 9 : Printf ("Elements not found")
Step 10 : else
Step 11 : Printf ("elements found at", loc)
Step 12 : end of loop
```

Data Vs Information →



Sub 1	Sub 2	Sub 3	Sub n
12	14	16	18
13	15	17	19

ARRAY → Array is the collection of homogeneous data types and store in sequence of adjacent memory blocks.

eclaration → A [LB : UB]
(lower bond) (upper bond)

ONE DIMENSIONAL ARRAY →

Q→1) A [95 : 102] . find length of an array.

Sol^n → size → (UB − LB) + 1 = (102 − 95) + 1

= 7 + 1 = 8 element store

Two DIMENSIONAL ARRAY →

Declaration → $A[LB_1 : UB_1][LB_2 : UB_2]$

Q→1:) A [92 : 107 · 32 : 51] find size

Sol^n → A [16 : 20]

HINT → (107 − 92) + 1 = 15 + 1 = 16

(51 − 32) + 1 = 19 + 1 = 20

Q→2:) Arr [5:7], find size and Base address of Arr[5] = 10051 then find address of A[6] and A[7]

Sol^n → Size = (7 − 5) + 1 = 2 + 1 = 3 bytes

A[5] = 10051

A[6] = 10054

A[7] = 10057

HINT → 10051 + 3 = 10054 then 10054 + 3 = 10057

Q→3:) Arr [5:71], and size of each element is 6 byte and A[5] = 10057 then find A[6] and A[59].

Sol^n →

A[5] = 10057

A[6] = 10063

A[59] = 54 × 6 = 324 → 10057 + 324 = 10381

HINT → A[5] = 10057 (given) then we add 6 in 10057 to find A[6]. then to calculate A[59] in easy way, first we Consider index of Arr start from 1 means we subtract 4 from 5 to start index from 0. Now, our aim is to calculate A[59], again we subtract 4 from 59, then it come 55, then again we subtract 1 from 55, then it comes 54. Now, size is 6, so we multiply 54 by 6 and it comes 324 and then we add 324 in 10057 and finally it gives 10381

Q→4) A[1:20], find A[4], A[1], A[2], A[3], A[15]

given → size = 5 & base address = 1000

Soln →

A[1] = 1000
A[2] = 1005
A[3] = 1010
A[4] = 1015

A[15] → 14×5 = 70 → 1000+70 ⇒ 1070

DEFINITION OF DATA STRUCTURE →

TYPES, OPERATIONS →

TIME - SPACE TRADE OFF →

HINT → Space ↑ Time ↓
space ↓ Time ↑.

COMPLEXITY OF AN ALGORITHM →

Complexity of an algorithm is the function which gives the running time and/or space in term of input size.

Scanned by CamScanner

```
e.g → for (i → 3)
      {
      for (i → 3)
      {
          body
      }
```

complexity = n^2

$n = 3$

$= (3)^2 = 9$

complexity = n = 3 → (one loop)

ALGORITHM →

Q → find the largest element in an array →

Sol^n →

a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]
4	-3	7	9	15	14	19	-5	6

```
Step 1 → Set  K = 1, loc = 1 , max = data [1]
Step 2 → Repeat Step 3 and 4 While K < N
Step 3 → If  max < data[K]
             then loc = K and max = data[K]
Step 4 → Set K = K + 1
Step 5 → Write loc , max
Step 6 →   Exit
```

or

$A[\overset{LB}{\overset{\uparrow}{5}}, 6, 9, 4, 3, \overset{UB}{\overset{\uparrow}{2}}]$

```
Step 1 →   Big = A [LB]
Step 2 → loop,  i = LB + 1 : UB  step by 1
Step 3 →          if (Big < A [i])
Step 4 →              Big = A [i]
Step 5 → end if
Step 6 → end loop
Step 7 → Print Big
```

Q → find the selond largest number in an arr

A = [5, 7, 9, 3, 2, 8, 11]

Sol^n

```
Big = A[LB]    ,   S_Big = A[LB+1]
loop, i = LB+1 : UB  step by 1
    if ( ~~Big~~ A[LB] > A[LB+1])
            Big = A[LB]
        S_Big = A[LB+1]
    else
          Big = A[LB+1]
        S_Big = A[LB]
loop, i = LB+2 : UB step by 1
     if (Big < A[i])
        S_Big = Big
         Big = A[i]
     else if (S_Big < A[i])
           S_Big = A[i]
Printf ("sec largest no. is", S_big)
```

TIME - SPACE TRADE OFF → By increasing the amount of space for storing a data, one may be able to reduce a time needed for processing the data, or Vice versa.

NOTE → priority given as →

++a , + , = , a++

Q → a = 10

Calculate K = a++ + ++a ;

Sol^n K = (a++) + (++a)

$$K = \underset{11}{\underset{\downarrow}{(a++)}} + \underset{11}{\underset{\downarrow}{(++a)}}$$

Scanned by CamScanner

$K = 11 + 11 = 22$

HINT → Ist step is ++a then its value is comes 11. a++ is also equal to 11, then we add both value, it come 22.

last priority = a++
= 12 (a = 11)

HINT → a = 11 + 1 = 12

Call By Value And Call By Refernce → Swap Programm And Sum Programm

Linear Search →

```
1. Set K = 1, loc = 0
2. Repeat Step 3 and 4 while loc = 0 and K <= n
3. If item = data[K] Then loc = K
4.    Set K = K + 1
5. If loc = 0 then write : item Not found
         else write : "loc" is the location of item
6. Exit
```

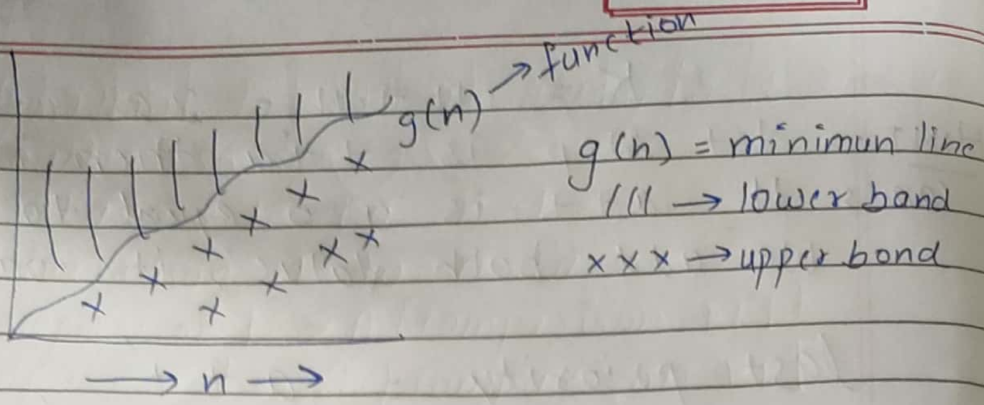
In programming → k = 0, loc = -1
first step → Read Array, Read item

10 or 2 marks

Asymptotic Notation →

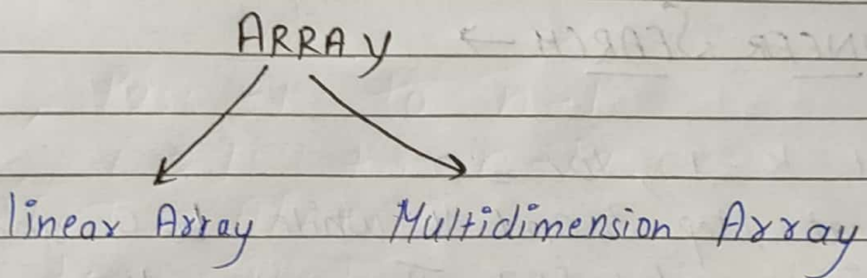
1.) Ω (omega Notation) → It gives the lower bond of the function.

2.) O (Big O Natation) → complexity → O(n)
It gives the upper bond of function.



3.) θ (Theta Notation) → It gives the value b/w lower bond and the upper bond.

ARRAYS →



1.) linear Array → linear Array is a list of a finite no. (n) of homogeneous data elements such that

a.) the element of an array referenced respectively by an index set consisting of n consecutive number

b.) the element of the array are stored respectively in successive memory location

★ TRAVERSING A LINEAR ARRAY →

```
1. Set K = LB
2. Repeat Step 3 and 4 while K <= UB
3. Apply process to LA[K]
4. Set K = K+1
5. Exit
```

ROHIT VAID SIR   ALGO

Q → UB [A : LB], find location 'roi' which is the bigg
element.

Solⁿ →
```
Step 1 → roi = A
Step 2 → loop, i = A+1 to LB step by 1
Step 2·a) → if (UB[roi] < UB[i])
Step 2·a·1) → roi = i
        end if
        end loop
Step 3 → Printf ("the biggest element UB[roi] is
            found at location", roi)
Step 4 → end
```

BUBBLE SORT →

Q → [5 9 2 6 7 1 4]. How to sort an array using bubble sort.

→ [5 9 2 6 7 1 4]

Pass I → [5 2 6 7 1 4 9]   n = 7
Comparison = 6

Pass II → [2 5 6 1 4 7 9]   Comparison = 5

Pass III → [2 5 1 4 6 7 9]   Comparison = 4

Pass IV → [2 1 4 5 6 7 9]   Comparison = 3

Pass V → [1 2 4 5 6 7 9]   Comparison = 2

Pass VI → [1 2 4 5 6 7 9]   Comparison = 1

NOTE → No. of Comparison = n − Pass

Scanned by CamScanner

Q → Write algorithm, to sort an array [5 9 2 6 7] using bubble sort.

→
```
loop 1 , Pass = 1 to n-1
loop 2 , comparison = 1 to n-Pass
if (A [comparison] > A [comparison +1])
    A [comparison] = A [comparison +1]
    A [comparison +1] = A [comparison] - A [comparison +1]
    A [comparison] = A [comparison] - A [comparison +1]
end if
end loop 2
end loop 1
return (A)
```

OR

```
loop 1 , Pass = 1 to n-1
loop 2 , i = 1 to n-Pass
if (A[i] > A[i+1])
        temp = A[i]
        A[i] = A[i+1]
        A[i+1] = temp
end if
end loop 2
end loop 1
return (A)
```

PROGRAMMING →

```
void main ()
{
Void bubble (int [], int);
int a[100], n, i ;
Printf ("Enter the length of an array");
Scanf ("%d", &n);
```

```
Printf ("Enter the element");
for (i=0 ; i <=n-1 ; i++)
Scanf ("%d", &a[i]);
bubble (a,n);
Printf ("sorted array is given below");
for (i=0; i<=n-1 ; i++ )
printf ("%d \t", a[i]);
}
void bubble (int a[] , int n)
{
int Pass , comporison , temp , i ;
for (i=0 ; i<=n-2 ; i++)
 for( comp =0 ;comp<=n-i ; comp ++)
     if (a[comp] > a[comp+1] )



end if
end loop 2
end loop 1
return (A)
```

S.P. SINGH SIR

★ INSERTION IN ARRAY →

```
Step 1 → Set J = ~~K~~ N
Step 2 → Repeat 3 and 4 While (J >= k)
Step 3 → Set LA[J+1] = LA[J]
Step 4 → Set J = J-1
Step 5 → Insert LA[K] = item
Step 6 → Set N = N+1
Step 7 → Exit
```

Scanned by CamScanner

given →

lA[1]	lA[2]	lA[3]	[4]	[5]	[6]	[7]		
4	6	9	-15	16	3	-19		

N = 7
K/ loc = 5
item = 7

output →

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	
4	6	9	-15	7	16	3	-19	

↑

N = 8

NOTE →

```
for (i = 0; i < n; i++)
{
printf (" a [%d] = %d ", i, a[i]);
}
```

★ DELETING A ELEMENT IN ARRAY →

```
Step1 → Set item = LA[K]
Step 2 → Repeat 3 for J = K to N-1
                      J = 3 to 6-1 = 5
Step 3 → Set LA[J] = LA[J+1]
Step 4 → set N = N-1
Step 5 → Exit
```

given →

1	2	③ ↓	4	5	6		
3	-5	6	9	10	11		

N = 6
loc / K = 3

output →

1	2	3	4	5		
3	-2	9	10	11		

N = 5

HINT →

J = 3	J = 4	J LA = 5
LA [3] = LA [4]	LA[4] = LA [5]	LA[5] = LA[6]

Scanned by CamScanner