PAGE NO.: 2 | 12

★ Assignment ★

① Define data & data structure, also discuss Uses and advantages of Using the data structure.

⇒ Data :

Row observations and values.

- The data structure is defined as a way of organizing data in such a way so that data can be Used efficiently (Space & time)

- Need for Data structure :

① Searching Large amounts of Data :

To retrive required data efficiently from the large amount of data generated and stored.

2) Speed of Processing :

Searching and retriving data from the well organized bunch takes less time and less effort.

3) Concurrent requests :

Many and simultaneous requests can be easily handled

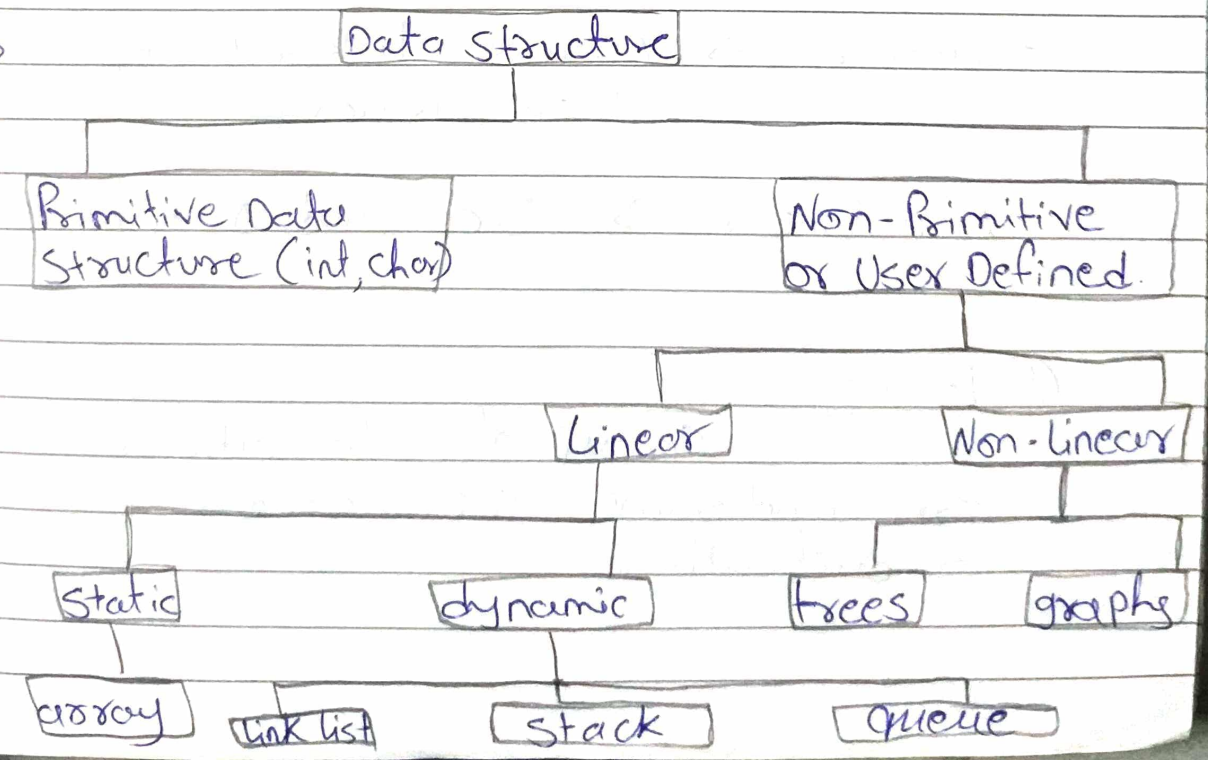
PAGE NO.:

→ Advantages of a data structure :

- Correctness :
  Correctly implemented
- Time Complexity :
  Running time or the execution time of operations of data structure must be as small as Posible
- Space Complexity :
  Memory usage of a data structure operation should be as little as Possible

2) Draw diagram of various types of data structure and explain each type in one line ?

→



① Primitive data structures :

Primitive data structure are basic structures and are directly operated upon by machine instructions.

- Primitive data structure have different representations on different computers.
- Integers, floats, character and Pointers are examples of Primitive data structure

② Non - Primitive Data type :-

These are more sophisticated data structure

- A Non-Primitive Data structure is further divided into Linear and Non-linear data structure.
- Linear → Array, Linked list, Stack, Queue.
- Non-Linear → Tree and graphs

① Array :

Group of data with same type and same size stored adjacent to each other.

- array are always stored in consecutive memory locations.
- It can be one dimensional or many dimensional.

2) Stack :

Stack is first in last out Structure.

- only top element can be accessed.
- for example, we can place or remove a card or plate from the top of stack only

- Basic operations in stack is Push () and Pop ().
- other operations is Peek (), isfull () and is Empty ().

3) Queue :-

Queue is first in First out.

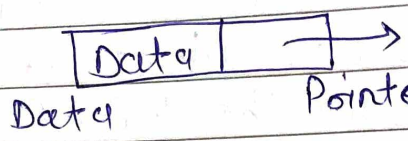
- AQueue can be implemented by means of Array, Structure, Pointer and Linked list.

- Basic operation in Queue is Insert () and delete ().
- other operations is Peek (), is full (), is Empty ()

4) Linked List :-

Linked list consists of linked nodes.

- Each node is having a data and a Pointer Part.
- The data Part stored data in it.
- The Pointer is an address Pointing towards the next element of the list.



Node

- Types of linked list :-
  1) simple linked list.
  2) Doubly linked list
  3) Circular linked list

⇒ Non-Linear :-

1) tree :-

Tree represent the nodes connected by edges.

- If each node is having maximum 2 connected nodes, then it is a binary tree.

- Basic operation on tree is, Insertion, Deletion, Traversal.

PAGE NO.:

2) graphs :-

graph is represented as a set of vetices (node or Points) connected by edges (arcs or line)

- graphs are used to represent network of cities, or telephone network or circuit network. and also used in social network like Facebook.

- Basic operations such as insertion, deletion, traversal, finding Path etc. are possible on the graph data structure.

3) List and explain in one line Various operations on data structure ?

⇒
1. Define or create structure.
2. Add an element
3. Delete an element
4. Traverse / Display
5. Sort the list of element
6. Search for a data element
7. Merging and spliting
8. Delete an element or delete complete structure.

1) Define or create Structure :-
Define and create a data structure according to the requirement

2) Add an element :-
Add or Insert a new data items in the data structure

3) Delete an element :-
Delete an existing data item from the data structure

4) Traverse / Display :-
Access each data item exactly once so that it can be Processed.

5) Sort the list of element :-
Arranging the data items in some order.

6) Search for a data element :-
Find out the location of the data item if it exists in data structure.

7) Merging and spiting :-
Combining the data items of two sorted file into single file in the sorted form.

-combining the data item spliting single data items from multiple Data items.

3) Delete complete Structure :-

Destroy the full data structure, existing data items and data element both Delete in data structure.

4) Differentiate Linear & Non-Linear data structure.

⇒ Linear Data Structure	Non-Linear data Structure
1) Every item is related to its Previous and next time	Every item is attached with many other items.
2) Data is arranged in Linear sequence.	Data is not arranged in sequence.
3) Data items can be traverse in a single run	Data can not be traversed in a single run.
4) E.g Array, Stack, Queue, Linked List	E.g tree, graph
5) Implementation is easy	Implementation is difficult.

5) Define linear data structure. Explain static & dynamic linear data structure ?

⇒ A data structure is said to be linear if its elements are connected in linear fashion by means of logically or in sequence memory locations

- two way to represent linear data structure in memory.

1 Static memory allocation
2 Dynamic memory allocation

1) Static memory allocation.

- Array :-

Group of data with same type and same size stored adjacent to each other.

- Array are always stored in consecutive memory location.
- Each memory location stores one fixed-length data item
- it can be on dimensional or many dimentional

PAGE NO.:

- one - dimensional Array of integer :

A[0]	A[1]	A[2]	A[3]
10	20	30	40

- here A is name of an array
- value in bracket are called index.
- Total no of element are 4.
- All are integers.

→ Two Dimensional Array of integer :

A [Row, Col]

Col →	0	1	2	3
Row				
0	A [0,0]	A [0,1]	A [0,2]	A [0,3]
1	A [1,0]	A [1,1]	A [1,2]	A [1,3]
2	A [2,0]	A [2,1]	A [2,2]	A [2,3]
3	A [3,0]	A [3,1]	A [3,2]	A [3,3]

2) Dynamic memory allocation :-

1) Stack :

Stack is first in last out structure. only top element can be accessed.

- for e.g, We can Place or remove a card or Plate from the top of the stack only

- Basic operations is Push (), and Pop ().
- Other operations is Peek (), isFull () and is Empty ().

2) Queue :-

Queue in first in first out.

- A queue can be implemented by means of Array, structure, Pointer and Link List.

- Basic operation is Insert () and Delete ().
- Other operation is Peek (), is Full (), and isEmpty ().

3) Linked List :-

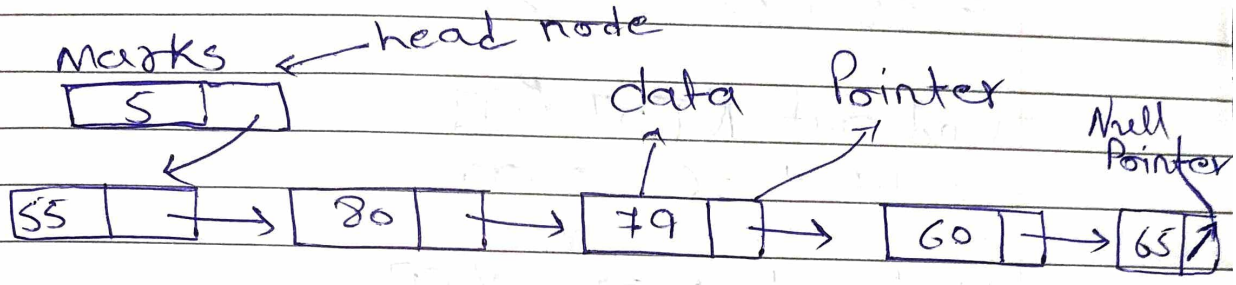
Link List consists of Linked nodes.

- Each node is having a data and a Pointer Part.
- The data Part stores data in it.
- The Pointer is an address Pointing towards the next element of list

A Node

- Each list is having a head node.
- In head node data part contains name of list and Pointer contain address of the first node of the list.

Example



Q Write a short note on array ?

⇒ Group of data with same type and same size stored adjacent to each other.

- Array are always stored in Consecutive memory location.
- It is a linear data structure with known number of element.
- Each memory location stored one fixed length data items.
- Array can Used in all Programming language
- It can be one dimensional or many dimensional array.

PAGE NO.

Example, one dimensional array:

N[0]	N[1]	N[2]	N[3]	N[4]
44	56	58	62	89

- Find out Num[2] + N[3]
  = 58 + 62 = 120

- Find out N[0] + N[1]
  = 44 + 56 = 100

- Find out N[3] - N[1]
  = 62 - 56 = 6

⇒ Two dimensional array;

0	0	1	2	3	4
1	'A'	'B'	'C'	'E'	'J'
2	'K'	'N'	'P'	'F'	'K'
3	'X'	'P'	'C'	'I'	'Z'
4	'Y'	'Q'	'D'	'H'	'X'

- Name [1,3] = 'F'
- Name [3,4] = 'Z'
- Name [4,1] = 'Q'
- Name [1,4] = 'J'

- Define Array in various Programming Language.

1	Java	Long arr[] = new long[5];
2	C	Long arr[5];
3	Python	arr = [None] * 5
4	Javascript	Var arr = [];

7) Define a one dimension array called weights of 5 Persons in real no. Calculate
1 minimum weight
2 maximum weight
3 average weight of the 5 Persons.

→

```
# include <iostream.h>
# include <conio.h>
# include <string.h>

int main ()

{
    clrscr ();
    const n = 5;

    float w[n], tot = 0, avg = 0, max = 0,
        min;
```

```
for (int i=0; i<n; i++)
{
  count << "\n Enter weight : ";
  cin >> w[i];

  tot = tot + w[i];

  if (max < w[i]) max = w[i];
  if (i==0) min = w[i];
  if (min > w[i]) min = w[i];
}

avg = tot/n;

cout << "\n w1 \t w2 \t w3
     \t w4 \t w5 \t max \t min
     \t avg ";

cout << "\n ========
     ===== \n";

cout << w[0] << "\t" << w[1]
     << "\t" << w[2] << "\t" w[3]
     << "\t" << w[4] << "\t" << max
     << "\t" << min << "\t" << avg;
```

```
    getch();
    return 0;
}
```

=> output :-

```
Enter weight : 30
Enter weight : 80
Enter weight : 79
Enter weight : 89
Enter weight : 110
```

w[0]	w[1]	w[2]	w[3]	w[4]
30	80	79	89	110

max	min	avg
110	30	77.6

8) Define two dimension array of your choice ?

⇒ Two dimensional array is an array within an array. It is an array of arrays.

- In this type of array the position of an data element is referred by two indices instead of one

⇒ Example Two dimensional array of Integers.

- A [Row, col]

col →	0	1	2	3
Row				
0	A[0,0]	A[0,1]	A[0,2]	A[0,3]
1	A[1,0]	A[1,1]	A[1,2]	A[1,3]
2	A[2,0]	A[2,1]	A[2,2]	A[2,3]
3	A[3,0]	A[3,1]	A[3,2]	A[3,3]

⇒

	0	1	2	3
0	'A'	'q'	'c'	't'
1	'x'	'x'	'D'	'f'
2	'y'	'z'	'K'	'i'
3	'P'	'O'	'L'	'J'

Name [1, 3] = 'f'
Name [3, 3] = 'J'

PAGE NO.:

Name [1, 2] = 'D'
Name [2, 3] = 'i'

Q) Write a short note on stack by explaining operations on the stack ?

⇒ Named stack as it behave like a real world stack, for e.g a deck of cards or a Pile of Plates, etc.

- First in Last out structure
- only top element can be accessed
- for e.g, we can Place or remove a card or Plate from the top of the stack only

- Stack can either be a fixed size one or it may have a sense of dynamic resizing
- Stack A stack can be implemented by means of array, structure, Pointer & Linked list.

⇒ Basic operations :-

- Push () :- Pushing (Storing) an element on the stack.

- Pop () :- Removing (accessing) an element from the stack

PAGE NO.:

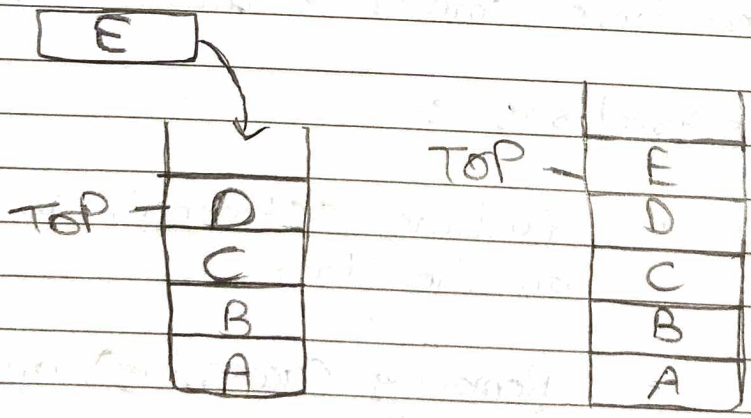
⇒ other operations :-

- Peek () : get the top data element of the stack, without remove it.
- isfull () :- check if stack is full.
- isEmpty () : check if stack is Empty.

⇒ Push operation :-

Putting a new data element into the stack.

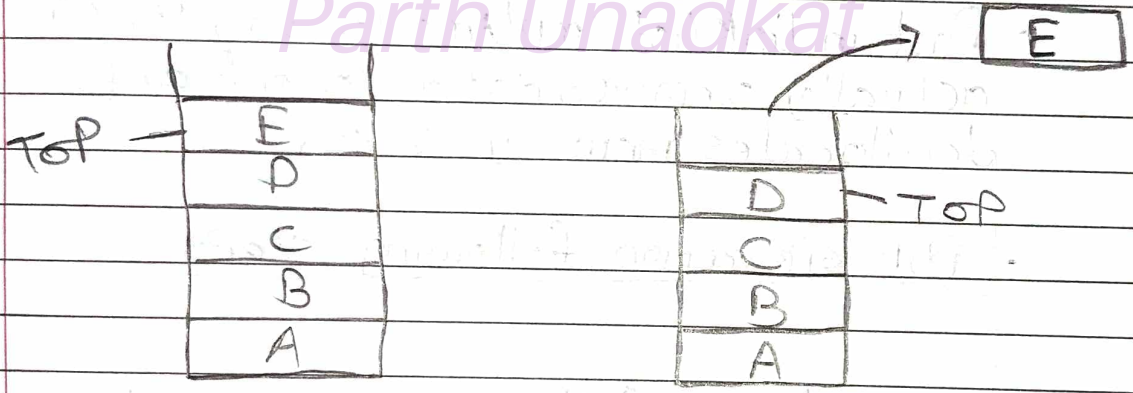
Step 1 : Check if the stack is full
Step 2 : If the stack is full, Produces an error and exit.
Step 3 : If the stack is not full, increment top to point next empty space.
Step 4 : Adds data element to the stack location, where top is pointing
Step 5 : Returns sucess.



→ Pop Operation :

Taking off the top data element from the stack.

Step 1 :- Check if the stack is Empty.
step 2 : If the stack is empty, Produces an error and exist.
step 3 : If the stack is not empty, accesses the data element at which top is Pointing.
Step 4 : Decreases the value of top by 1
Step 5 : Returns success.



→ Uses of stack :

- Parsing expression (infix, Prefix, and Postfix conversion)
- Recursion
- Flow of control and function call
- Back tracking Procedures and games

10) Write an outline of Pop operation in a stack !

⇒ Accessing the content while removing it from the stack is known as a Pop operation.

- In stack implementation of Pop() operation the data element is not actually removed instead top is decremented to a lower position in the stack to point to the next value.

- But in link list implementation, Pop() actually removes data element and deallocates memory spaces.

- Pop operation following steps :-

Step 1 : check if the stack is empty

Step 2 : If the stack is empty, Produces an error and exit

Step 3 : If the stack is not empty, accesses the data element at which top is Pointing

Step 4 : Decreases the value of top by 1

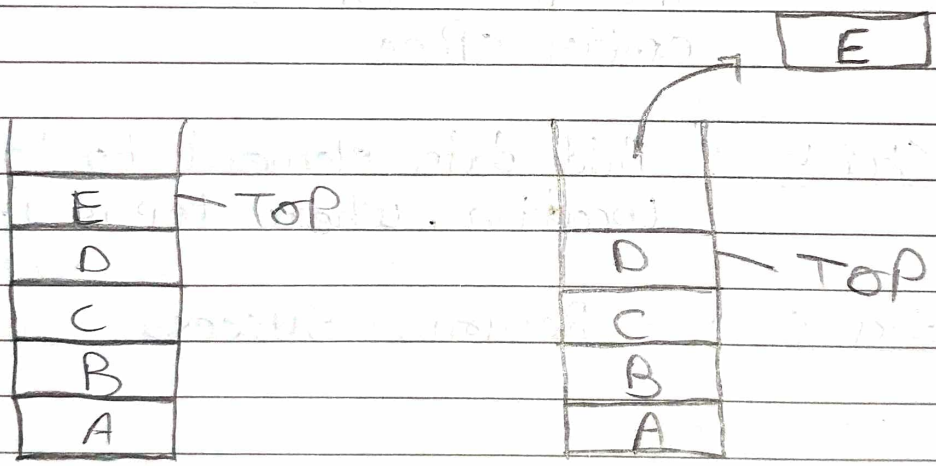
Step 5 : Return success

PAGE NO.:

- A simple algorithm for PoP operation can be derived as follow ;

- begin Procedure PoP : stack

```
if stack is empty
  return null
endif

data ← stack [top]
top ← top - 1
return data

end Procedure
```

11) Write an outline of Push operation in a stack ?

⇒ The Process of Putting a new data element on to stack is known as a Push operation.

- Push operation involves a steps ;

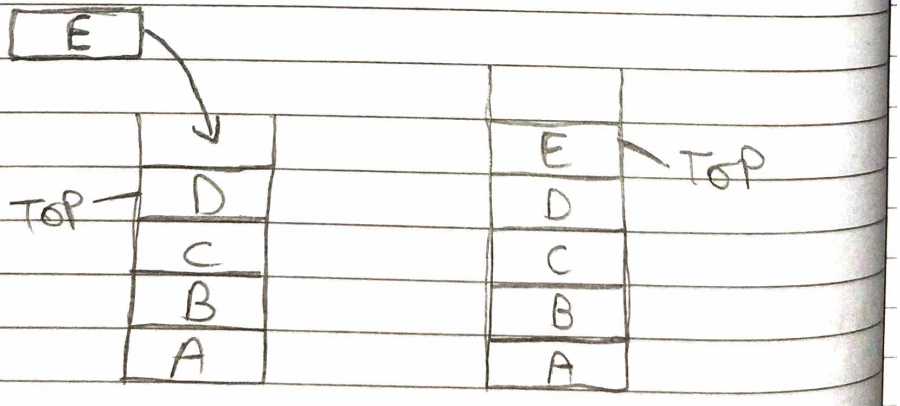
Step 1 : Check if the stack is full.

Step 2 : If the stack is full, Produces an error and exit.

Step 3 : If the stack is not full, increment top to Point next empty Space.

Step 4 : Add data element to the stack location, where top is Pointing.

Step 5 : Returns success.



- A simple algorithm for Push operation can be derived as follow;

```
begin Procedure Push : Stack, data

    if stack is full
        return null
    endif

    top ← top + 1
    Stack [top] ← data

end Procedure.
```


12) Write a short note on queue by explaining operation on the queue ?

=> Queue is an abstract data structure. Queue follows first in first out.

i.e the data item stored first will be accessed first.

- A real world example of queue can be one-way road, where the vehicle enter first, exit first.

- A queue can be implemented by means of Array, Linked list, Pointers, and Structures.

* Basic operations in queue :

- Insert () : always at the end
- delete () : always from the front

* other operation in queue :

- Peek () : get the data element from the queue, without remove it.
- isfull () : check if the queue is full
- isEmpty () : check if the queue is empty.

1) Insert operation :- inserting data in the queue at the end / rear position.

Step 1 : check if the queue is full.

Step 2 : If the queue is full, Produce overflow error and exit.

Step 3 : If the queue is not full, increment rear pointer to point the next empty space.

Step 4 : Add data element to the queue location where the rear is pointing.

Step 5 : return success

⇒ Variation on queue :

1) Priority queue

2) Circular queue.

2) Delete operation : (Deleting data at the beginning / front position)

Step 1 : check if the queue is empty.

Step 2 : If the queue is empty, Produce underflow error and exit.

Step 3 : If the queue is not empty, access the data where front is pointing.

Step 4 : Increment front pointer to point to the next avialable data element

Step 5 : Return success.

→ Uses of queue :

1) Operating system and resource management such as CPU, memory scheduling, Printer queue etc.
2) Call centre Phone systems
3) Scheduling jobs.
4) Searching

13) Write an outline of insert operation in a queue ?

⇒ Queue maintain two data pointers, front and rear. Therefore, its operation are comparatively difficult to implement than that of stacks.

- following steps taken to insert data into a queue.

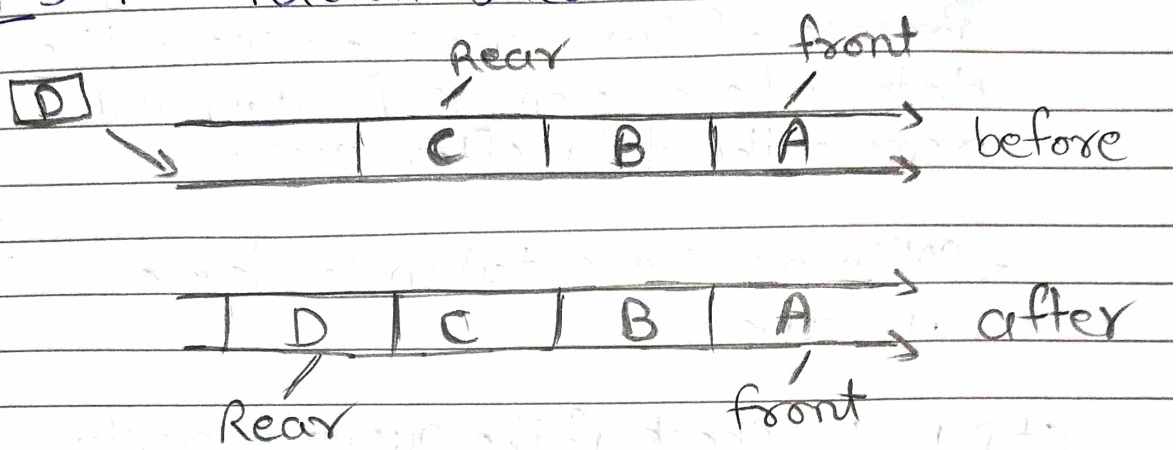
Step 1 : Check if the queue is full.

Step 2 : If the queue is full, Produce overflow error and exit.

Step 3 : If the queue is not full, increment rear Pointer to Point the next empty space

step 4 : Add data element to the queue location, where the rear is pointing

step 5 : return success.



- Sometimes, we also check to see if a queue is initialized or not, to handle any Unforessen situations.

- Algorithm for insert operation :

```
Procedure insert (data)

    if queue is full
        return overflow
    end if

    rear ← rear + 1
    queue [rear] ← data
    return true

end Procedure.
```

14) Write an outline of delete operation in a queue ?

⇒ Accessing data from the queue is a process of two tasks - access the data where front is pointing and remove the data after access.

- following steps taken to delete operation :-

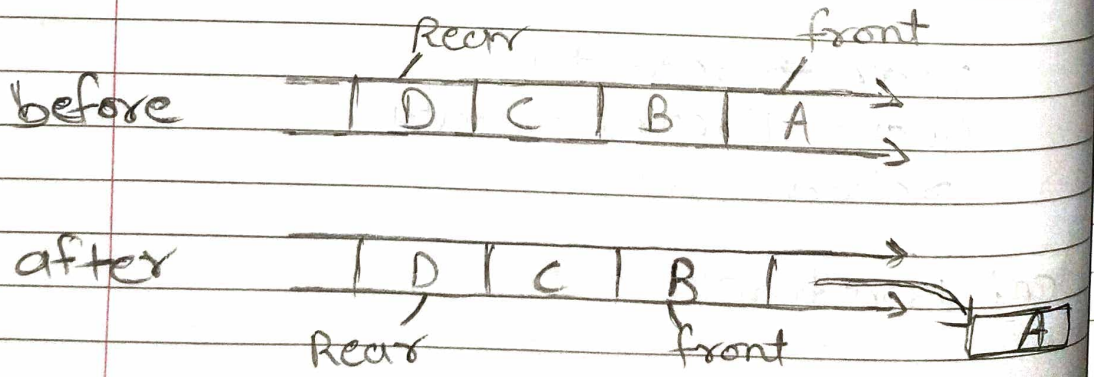
Step 1 : Check if the queue is empty.

Step 2 : If the queue is empty, Produce underflow error and exit.

Step 3 : If the queue is not empty, access the data where front is pointing

Step 4 : Increment front pointer to point to the next available data element

Step 5 : Return success.

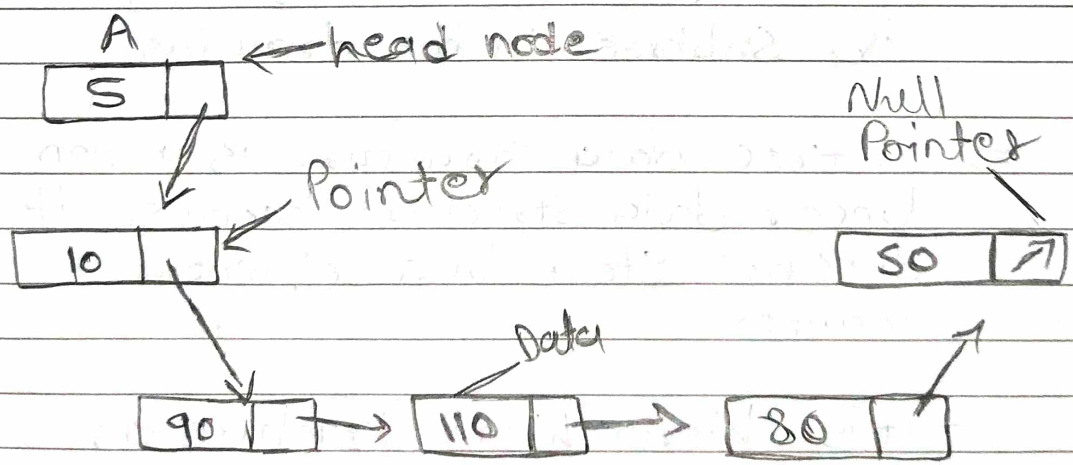


→ Algorithm for delete operation :

```
Procedure dequeue (delete)

    if queue is empty
        return underflow
    end if

    data = queue [front]
    front ← front + 1
    return true

end Procedure
```


15) Create a linked list of 5 integers called A ?

⇒ Linked list is a sequence of data structure which are connected together via links.

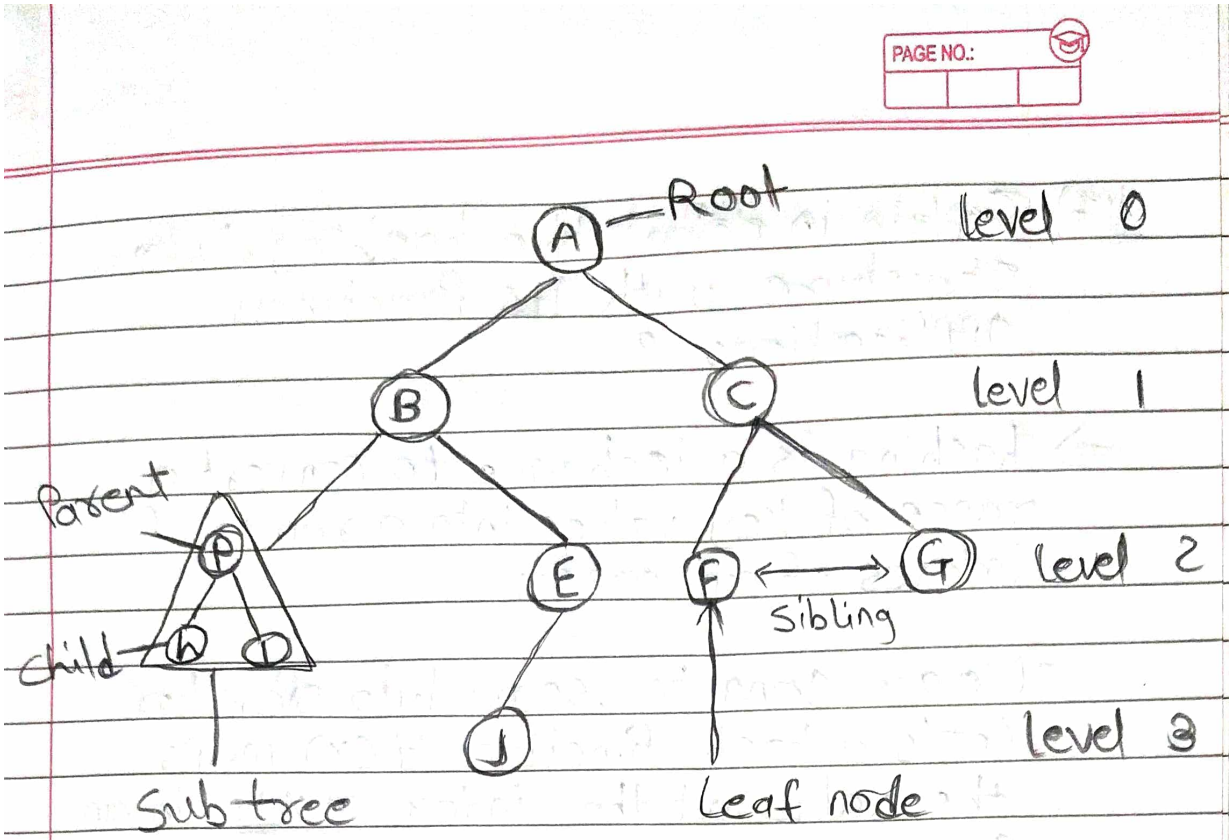


- In linked list A called 'Head node'
- A linked list create a 5 integers
- Each node is having a data and Pointer Part.
- The data Part stores data and Pointer Part Pointing toward the next element of the list.
- integer 66 is Pointer to the next integer 79.
- Last node in integer 41 and Pointer define Null Pointer.

16) Draw an example of tree data structure and show (i) root (ii) branches (iii) leaves and (iv) subtree in the drawing.

⇒ A tree data structure is a non-linear data structure because it does not store in a sequential manner

- It is a hierarchical structure as elements in a tree are arranged in multiple levels.



- In the tree data structure the topmost node is known as a root node.

- The arcs between nodes are called as branches. a node that has no branches underneath it called a leaf.

- In data structure node with no children are called leaves or external nodes.

- Subtree is represents the descendants of a node, the subtree corresponding to any other node is called a Proper subtree.

(7) Explain in detail the hashing data structure with its Practical applications ?

⇒ hashing is a technique to convert a range of key values into a range of indexes of an array.

- We are going to use modulo operator.
- Let's a hash function H(x) maps the value at the index x % 10 in an array

- for example if the list of value is [11, 12, 13, 14, 15] it will be stored at position {1, 2, 3, 4, 5}.

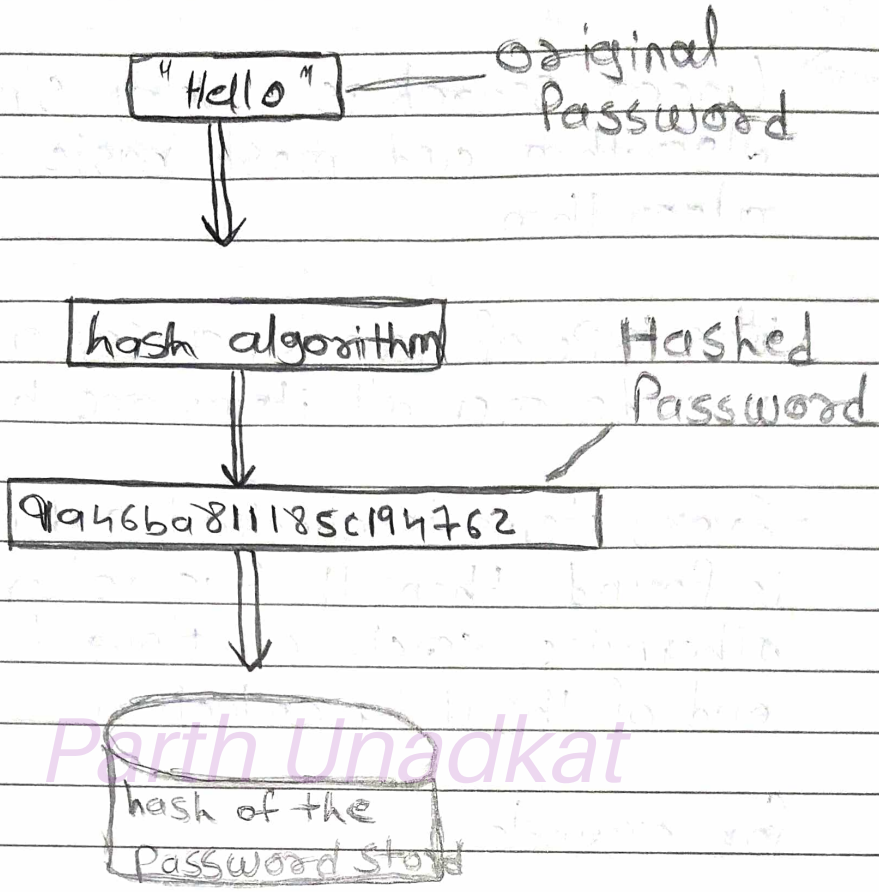
example List :- 11, 12, 13, 14, 15

x % 10

11%10 12%10 13%10 14%10 15%10

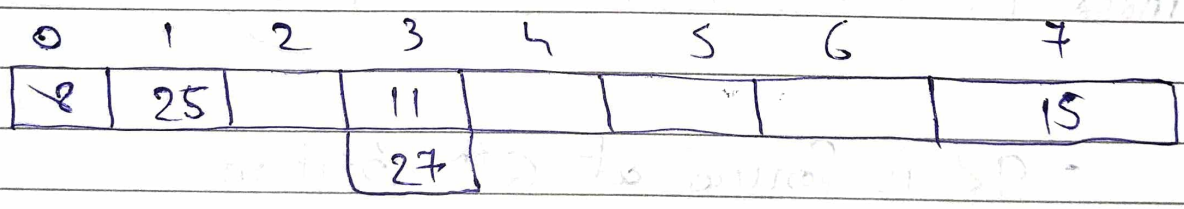
	0	1	2	3	4	5
H.T		11	12	13	14	15

- Application hashing Password :-



Example :

- 7 Buckets (0, 1, 2, 3, 4, 5, 6, 7)
- Data - (15, 11, 27, 8, 25)



18) Explain Linear search in detail by taking an example ?

⇒ Linear search is a very simple search algorithm. and most basic searching algorithm.

- In this type of search, a sequential search is made over all items one by one.

- Every items is checked and if a match is found then that is returned, otherwise search continue till the end of the data collection

for example ;

- Element to search :- 98

array index	0	1	2	3	4	5	6	7	8
	10	9	27	91	37	88	98	11	19

- 98 is found at 6th Position.

- Algorithm :

- Linear search (Array A, value x)

Step 1 : set i to 1
Step 2 : if i > n then go to step 7
Step 3 : if A[i] = x then go to step 6
step 4 : set i to i+1
Step 5 : Go to step 2
Step 6 : Print element x found at index i and go to step 8
Step 7 : Print element not found.
step 8 : Exit.

- Procedure Linear_search (List, Value)

```
for each item in the list
    if match item == value
        return the item's location
    end if
end for

end Procedure
```

eg

10	11	88	92	52	18	98	91	2
0	1	2	3	4	5	6	7	8

search element : 18 (5th Position)

PAGE NO.:

19) Explain binary search in detail ?

⇒ Binary search is an efficient algorithm for finding an item from a sorted list of items.

- binary search works by repeatedly dividing in half.

Example :-

Let us assume that we need to search the location of value 30 using binary search.

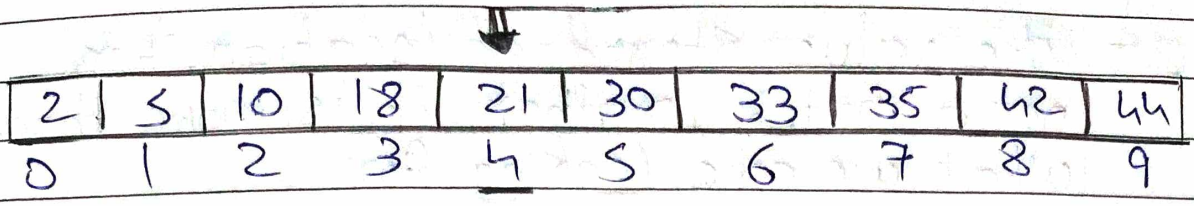
2	5	10	18	21	30	33	35	42	44
0	1	2	3	4	5	6	7	8	9

- first, we shall determine half of the array by using this formula,

mid = low + (high - low) / 2
here, 0 + (9 - 0) / 2 = 4 (floating value 4.5)

- So, 4 is mid value of the array.

- Now, we compare the value stored at location 4 the value begin searched, i.e 30.



- The value at Location 4 is 27, which is not a match.

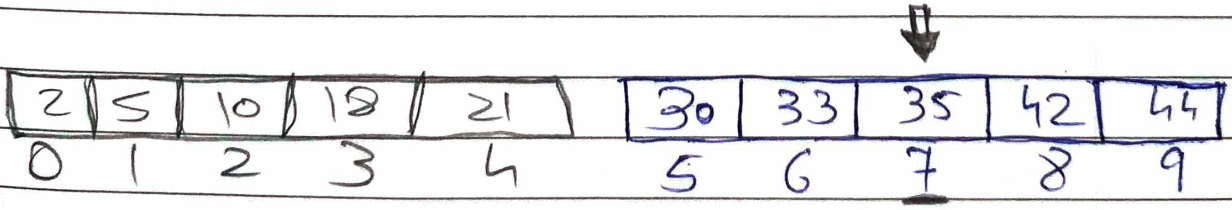
- As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper Portion of the array.

- The current situation like this.

- We can our low to mid + 1 and find the new mid value again.

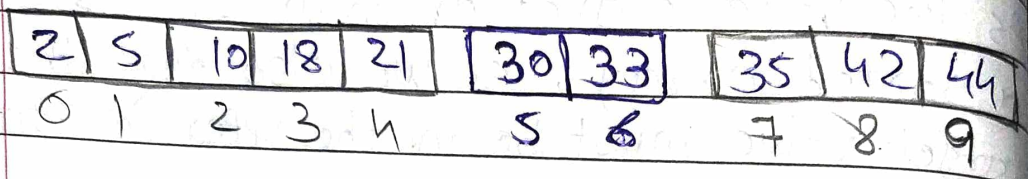
- Low = mid + 1
- New mid = low + (high - low) / 2
- our new mid is 7

- Compare the value stored at location 7 with our target 30.



- The value stored at location 7 is not a match, rather it is more than what we are looking for.

- So, the value must be in the lower part from this location.



- hence, we calculate the mid again This time mid value is 5.

- Compare the value stored at location 5 with our target 30. we find that it is a match

PAGE NO.:

20) Give main difference between binary search and linear search.

⇒

binary search	linear search
1 binary search is complex as compare to linear search	linear search is simple & straightforward to implement than the binary search.
2) binary search speed is fast.	linear search speed is slow.
3) Work well with arrays and not on linked list.	Work with arrays and linked list.
4) binary search in time complexity is O(log(n))	Linear search in time complexity is O(n).
5) binary search in condition is sorted.	Linear search in condition is any random order.
6) only single dimensional array is used.	multidimensional array can also be used.

PAGE NO.:

2) Explain bubble sort by taking an example ?

⇒ bubble sort is the simplest algorithm that worked by repeatedly swapping the adjacent element if they are in wrong order.

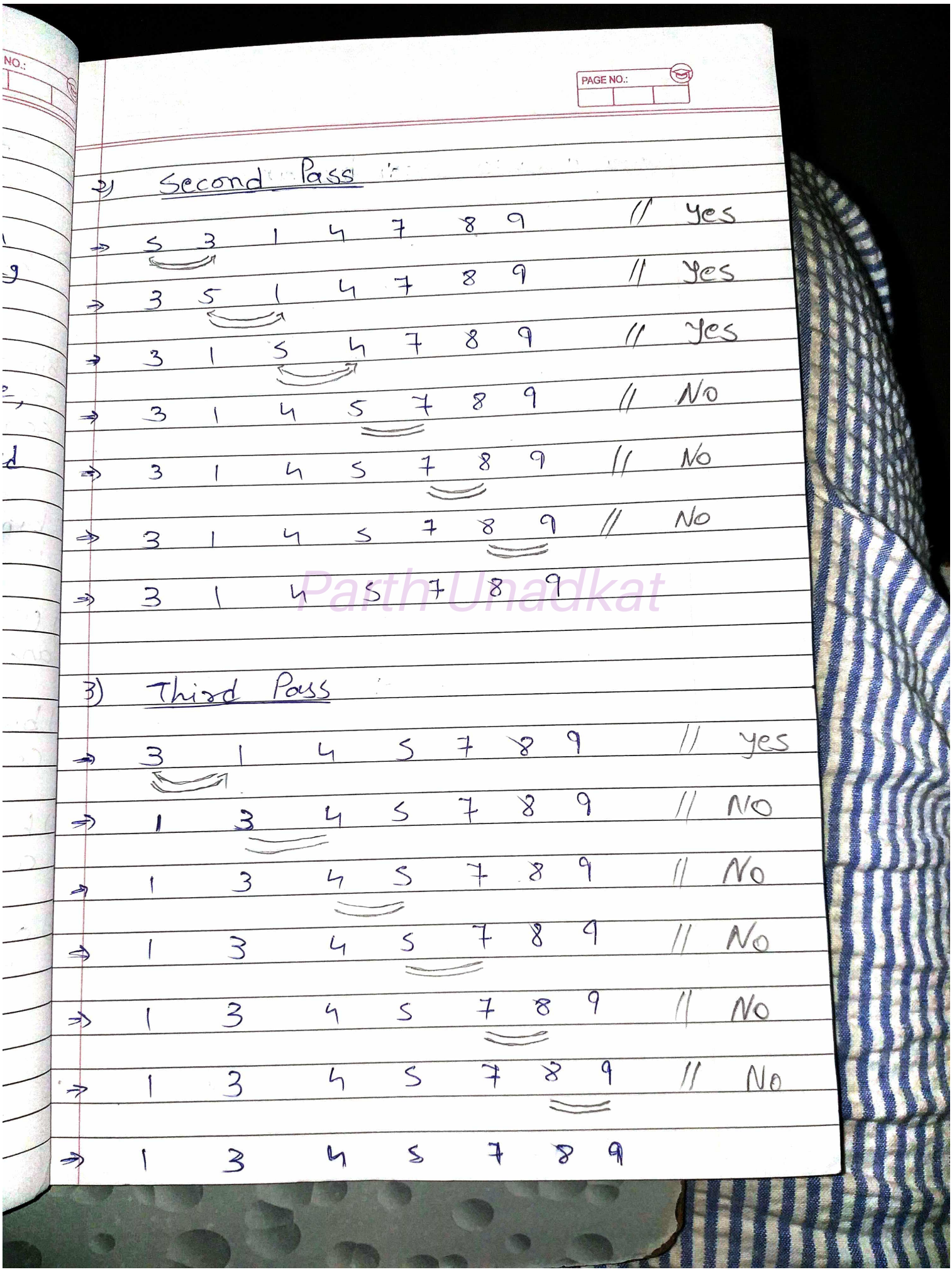
- Bubble sort is majorly used where,
  - complexity does not matter
  - simple and shortcode is Preferred

for Example :-

- 8 5 3 1 4 7 9

1 First Pass :

- 8 5 3 1 4 7 9 // yes
- ⇒ 5 8 3 1 4 7 9 // yes
- ⇒ 5 3 8 1 4 7 9 // yes
- ⇒ 5 3 1 8 4 7 9 // yes
- ⇒ 5 3 1 4 8 7 9 // yes
- ⇒ 5 3 1 4 7 8 9 // No
- ⇒ 5 3 1 4 7 8 9



2) Second Pass :

→ 5 3 1 4 7 8 9 // Yes
→ 3 5 1 4 7 8 9 // Yes
→ 3 1 5 4 7 8 9 // Yes
→ 3 1 4 5 7 8 9 // No
→ 3 1 4 5 7 8 9 // No
→ 3 1 4 5 7 8 9 // No
→ 3 1 4 5 7 8 9

3) Third Pass :

→ 3 1 4 5 7 8 9 // yes
→ 1 3 4 5 7 8 9 // No
→ 1 3 4 5 7 8 9 // No
→ 1 3 4 5 7 8 9 // No
→ 1 3 4 5 7 8 9 // No
→ 1 3 4 5 7 8 9 // No
→ 1 3 4 5 7 8 9