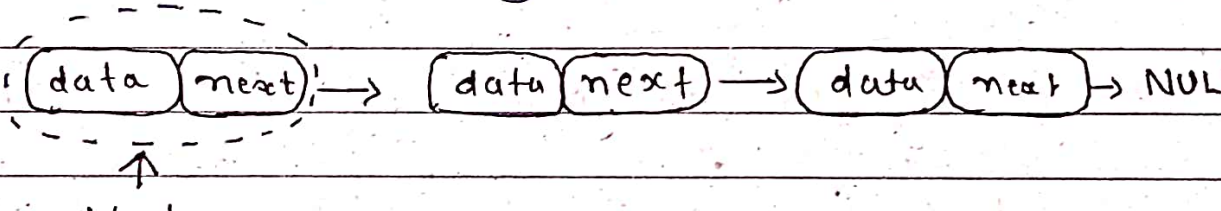
Linked List

→ Definition :- A Linked List is collection of object that is stored in list form.

→ Nodes :- Elements present in linked list called as...
- contain 2 things
  ① data.
  ② Pointer to the next node.

→ Linked List is linear data structure in which element are not stored in contiguous memory locations.



→ Data holds actual value associated with node.

→ Next pointer stores memory address of next node

→ LL accessed through head node, which points to the first node. The last node in the list points to NULL or nullptr indicating end of list. This node is called as tail node

→ Why LL Data structure ? Allows dynamic insertion and deletion of elements during run time.

→ Due to Ease of insertion and deletion compared to arrays, adjustments are minimal. With LL there is no requirement to shift elements, only the address needs updating.

→ LL can GROW as elements are added. LL can also SHRINK as elements are removed.

→ NO WASTAGE of MEMORY : because of fixed size allocation.

Example

In a system, maintaining sorted array such as id[] = [1000, 1010, 1050, 2000, 2040] can result in expensive operations. When inserting a new ID like 1005, elements need to be shifted, impacting efficiency. Similarly deletion also hinder the overall efficiency of code.

* Types of Linked List

3 types: Single Linked List
Double Linked List
Circular Linked List

① Singly Linked List

→ Linear data structure that holds value and reference to next node.
→ This list has head and tail.
→ We can traverse this LL in forward direction

Application

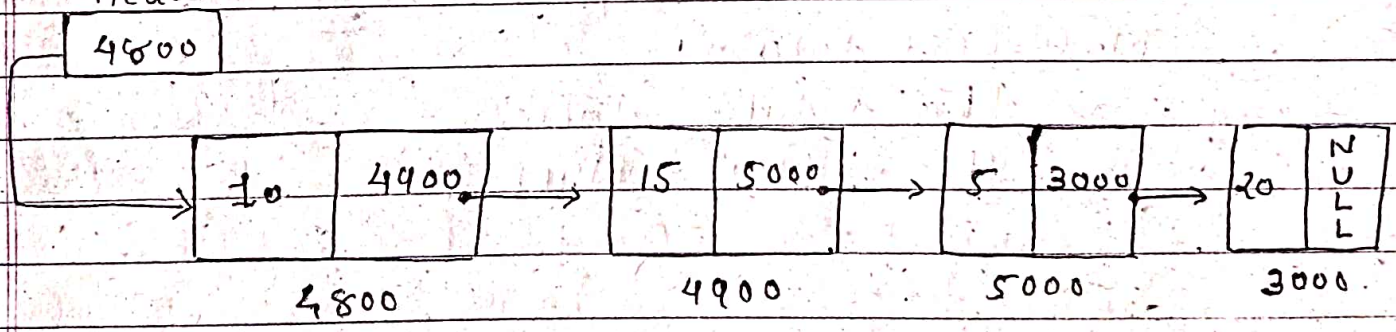
① memory management: enabling dynamic memory allocation and deallocation.
② Database indexing: Implemented in database for swift operations like insertion/deletion.

③ Representing polynomial and sparse matrix

④ In Operating Systems for tasks such as process scheduling and resource management.

Advantages
→ Dynamic memory allocation
→ Cache friendliness
→ Space efficient

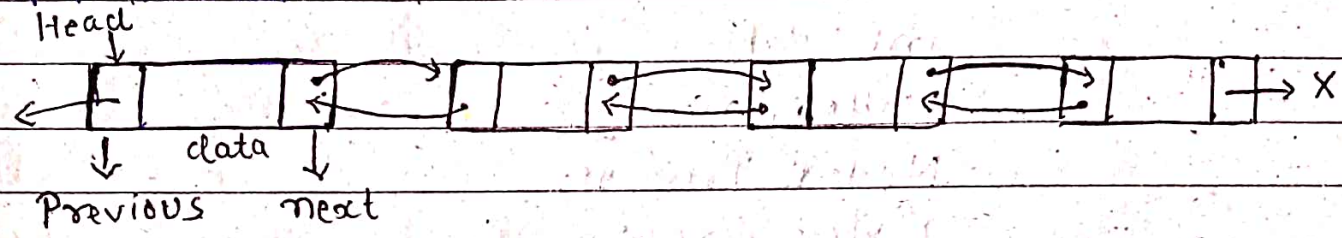
Disadvantage
→ Increased memory overhead
→ If nodes next pointer is lost, accessing elements is difficult.
→ Not suitable for parallel processing
→ Backward traversing not possible



SINGLY LINKED LIST

② Doubly Linked List

→ A Doubly Linked List (DDL) is a special type of linked list in which each node contains a pointer to a previous node as well as the next node of linked list.



* Application

- Used by web browser for backward and forward navigation of web pages.
- Most recently use or Least recently used cache are constructed using DLL.
- To maintain undo and redo functionalities.

* Advantages

- traverse in both direction.
- insertion and deletion of node is easy.
- DDL get the previous node so we can easily delete the node.

* Disadvantages

- requires extra space for previous pointer.

Stacks

→ Stack is type of linear list where you can insert and delete elements, but only at one end.

→ When you want to add element to stack it is called as PUSH. When you want to remove element from stack, it is called POP.

→ In a stack, the most accessible element called as top and the least accessible element called as bottom.

→ Due to work with one end of the stack, the elements are removed in opp order from how they were added. This behavior known as LIFO.

→ Whenever you add a new element to stack, you increase TOP pointer by "one". When you delete an element, the TOP pointer decrease by "one".

• operations on Stack

① Primary stack operation
- PUSH
- POP

② Auxillary stack operation
- TOP()
- size()
- isEmpty()
- isFull()

Date
Page

Application of Stack in DS.

- Recursion :- Recursion is like a function that keeps calling itself to solve problem in smaller steps. This stacks can remember these steps so you can go back when you are done
- Undo/Redo operation
- Expression evaluation: stacks make sure that you are calculating math problem in right order.
- Reversing characters
- Servicing Hardware interrupts
- Solving combinational problems, using backtracking

In real life:
- CD/DVD stand
- Stack of books
- Undo/Redo in text editors
- History of web browser
- Call logs, Google photos or Emails

Advantages of Stack
- Easy implementation
- Efficient memory
- Fast access time
- Supports backtracking
- Used in compiler design.

Disadvantages
- Limited capacity
- No random access
- Recursive function call limitations
- Stack overflow and underflow

(*) Operations on stacks

① PUSH OPERATION

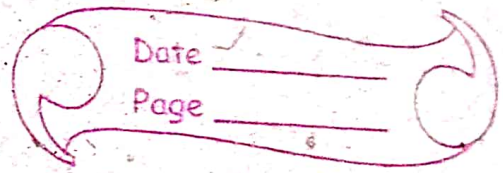
```
void push (int data){
if (SIZE - top > 1){
top++;
arr[top] = data;
}
else{
cout<<" SPACE NOT AVAILABLE "<<endl;
```

② POP OPERATION

```
void PoP() {
if (top == -1){
cout<<" Stack underflow "<< endl;
}
else{
top--;
}
}
```

③ PEEK OPERATION

```
int getTop()
if(top == -1){
cout<<"There is no element in stacks"<< endl;
return -1;
```




```
}
else{
 return  arr[top];
}}
```

Queue

→ Queue is defined as Linear Data structure in which all additions & deletions of the elements made at one end and all deletions from the list are made at other end.

→ It follows "First in First out" (FIFO) rule. The first item added is first one to be removed.

→ The end from where you remove items from the queue is called **front** (sometimes head of queue)

→ The end where you add new items to queue is called as **Rear** (sometimes tail of queue)

→ Adding item to queue called **Enqueque**

→ And, Taking an item out of queue called **Dequeue**

CHARACTERISTICS

- handles multiple data
- We can access both ends
- They are fast and flexible

Application

- multiprogramming :- Simultaneous execution of multiple programs
- Network : Data routing and communication infrastruct
- Job scheduling : Sequential execution of scheduled tasks assigned to processor

Real time application

- ATM Boothline
- Ticket counter line
- CPU task scheduling

Date
Page

Advantages

- efficient data management
- FIFO operations provide simplicity.
- Multiple - consumer service utilization
- swift interprocess communication

Disadvantages

- Slow middle operation
- Space Limitations
- O(N) search complexity
- Predefined maximum size.