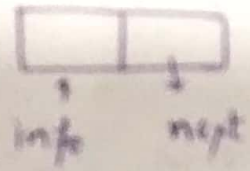
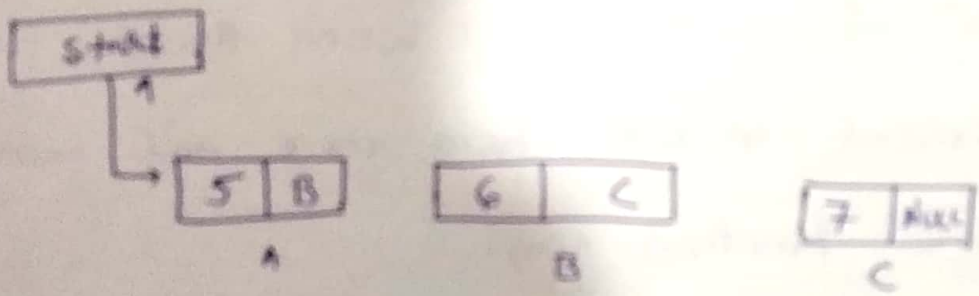
Linked list

linked list is a linear collection of data elements (also called nodes) in which each data element point to the next element.

The nodes have two or three parts.



The first parts contain information called info and another part contains address of next node is called next or link.



Type of linklist :— It is basically four types -

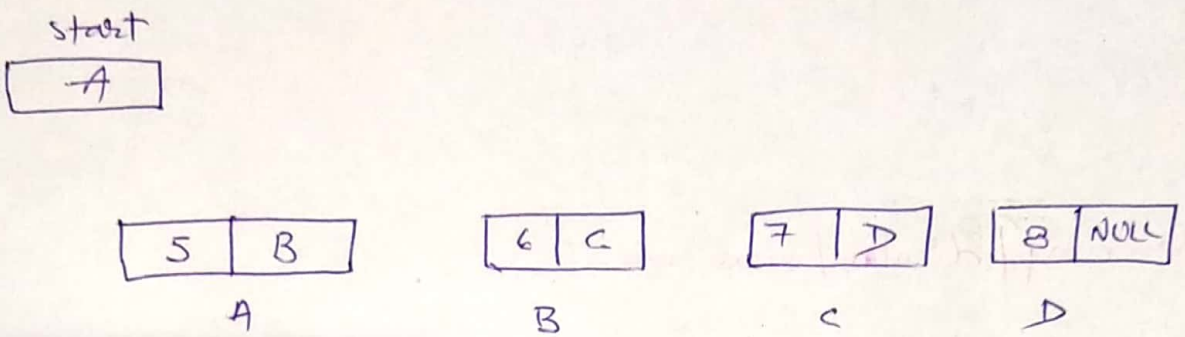
i) singly l l
ii) circular l l
iii) doubly l l
iv) doubly Circular l l.

Operations on l l

i) creation
ii) Insertion
iii) Deletion
iv) display/Traversing/view
v)

i) Single link list :—

In the single l.l, we can travel only one side.



Operation on single link list :—

① Create :— creation a link list in one way.

Algo :— create (START)

Let START is the node pointer that represent the starting of the list.

step 1 :- Set N = Assign [memory assign to new node N]

2 :- If N = NULL Then [check for memory allocation]
write: Overflow and exit

3 :- Read :- N → data (value) [input value to N]

4 :- Set N → next = NULL

5 :- If start == NULL then [check for empty list]
set Start = N and return
else
Set A = START

Repeat while A → Next != NULL

set A = A → next [End of while]

Set A ⇒ next = N

[End of if]

step6;- Exit.

2) Traversing / display / view ;—

Display (START)

Let START is node pointer that represent the starting list.

step1:- if START = NULL Then

write "empty list" and exit

step2;- set A = START

step3;- Repeat while A != NULL

write:- A → Value

set A = A → next

[End of while loop]

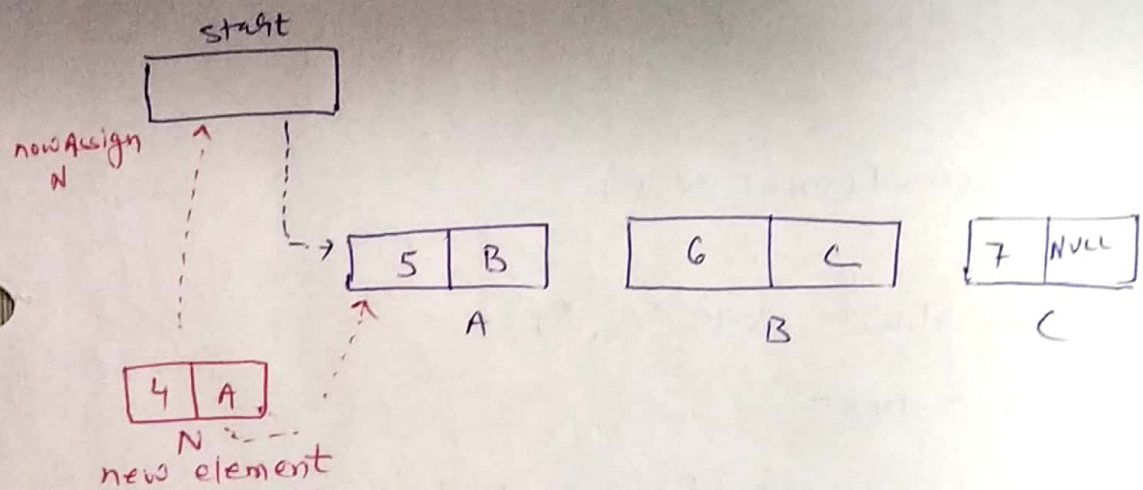
Step 4 ;- Exit

3) Insertion;— we can insert a ~~the~~ node in the L.L is 2 type.

i) begning of the list

ii) middle of or end of the list.

i) Begining of the list;—



Algorithms;—

Insertion at begining (START)

step-1 Set START = AVAIL

step-2 If ~~Node~~ node START = = NULL Then
Print overflow and exit.

step-3 Declare Struct Node *n

step 4 set n = Assign memory address using malloc function
[n, = (node*) malloc (size of (node))];

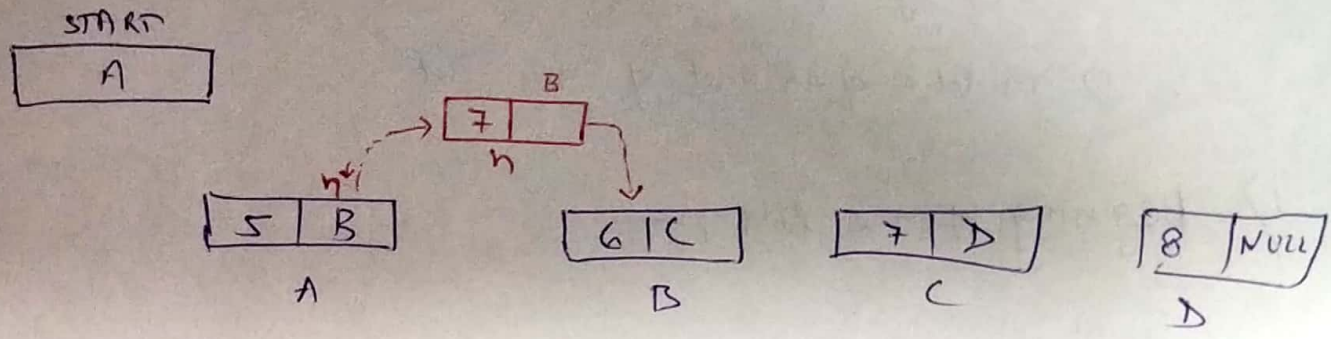
step-5 set n → info = value

6 set n → next = START

7 set START → n

8 Return

b) Insertion in Mid or last ;—



Algorithms ;—

insertion at mid (START, Q, T)

step1 Begin

2 Declare struct Node *Q, *T

3 set Q = START

4 Repeat step a & b for I = 1 to position-1 by +1

a) set Q = Q → next

b) If Q = NULL, Then

write: " There are less than elements " and return

[End of loop]

step5 set T = assign memory address using malloc function

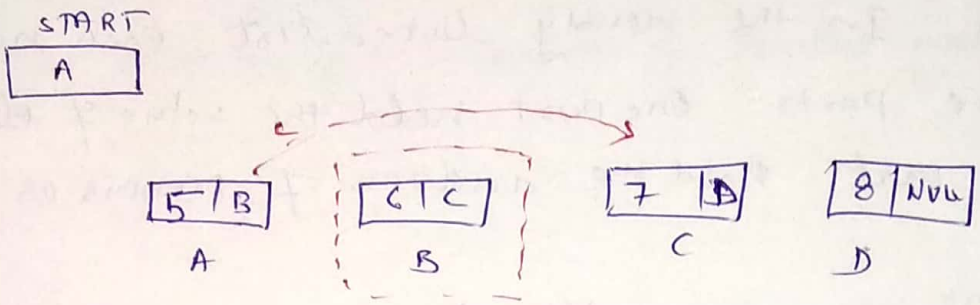
step6 set T → info = value

step7 set T → next = Q → next

step 8 set Q → next = T

step 9 Return

4) Deletion :—



Algorithms ;—

~~Delete~~ DEL (Info, next, START, AVAIL, LOC, LOCP)

This algorithms delete the node N with location LOC. LOCP is the location of the node which precedes N or when N is the first node, LOCP = NULL

Step 1 If LOCP = NULL then
set start = ~~next~~ start → next { delete first node }
else
set LOCP → next = loc → next [delete node N]
[End of If structure.]

step-2 ;— Set LOC → next = AVAIL
and AVAIL = LOC

Step 3 ;— Exit