BASICS:-

Pointers of arrays, strings and structures gives the address of first element or member.

Structures:

It is a user-defined datatype.

Name of Structure starts with a capital.

Structure concept came to overcome the array.

Typedef is used create alias names.

Declaration of a structure:

```
struct Student{
    int total;
    float per;
    char name[100];
}
```

Creating Alias Names for structure:

typedef struct Student STD;

Initialization of variables:

STD s1;

Disadvantages of function:-

Returns only one element at a time.

To overcome this we use the concept of arrays and structures.

Stack , Queue and Linked Lists are Linear data structures.

Trees and Graphs are non-linear data structures.

```
Linked Lists:- 4 types
```

Single LL

Circular Singular LL

Double LL

Circular Double LL

Linked List is collection of nodes.

Node is divided into two parts.

	Address					
data	of next					
	node					

In Linked Lists, address part of last node is always NULL which indicates the end of the linked lists.

Node Structure:

```
struct Node{
  int data;
  struct Node *next;
};

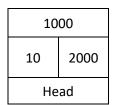
typedef struct Node NODE;
NODE n1,n2,n3;
```

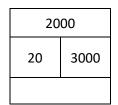
Linked Lists:-

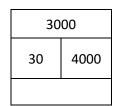
When Linked List is empty. Both Head and Tail are at NULL.

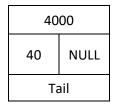
Linked List -> empty => head =NULL and tail = NULL

Traversing means travelling from first node to last node.









Basic operations of Linked List:-:-

Algorithms of Insert, Delete and Display

General terms in all algorithms:
NN = NewNode

res = result or value

Basic Operations in LL:

Insert Delete Display

Insert :-

Empty LL

When LL is empty, copy NN to head and tail i.e;

head = NN;

tail = NN

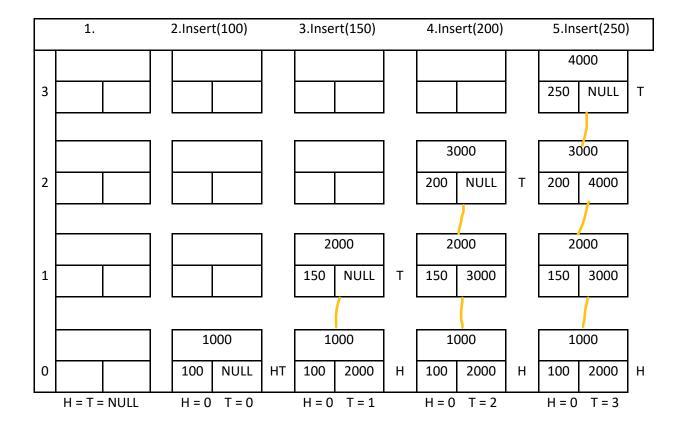
General condition

Copy NN to tail->next i.e; tail->next = NN and copy NN to tail i.e; tail = NN

In the below table these are the steps for inserting different nodes :-

- 1. Since there are no nodes in a Linked List, Both head and tail are at NULL which indicates LL is empty.
- 2. Here we have inserted 100 with address 1000. Since there is no any other next node for 100. It's next node address is taken as NULL and there are no any link to it.
- 3. Here we have inserted 150 with address 2000. And next node's address of 100 as changed to 2000 to create a link between 100 and 150. Since there no other next node for 200. It's next node address is taken as NULL.

4 & 5 steps follows same as the 3rd step.



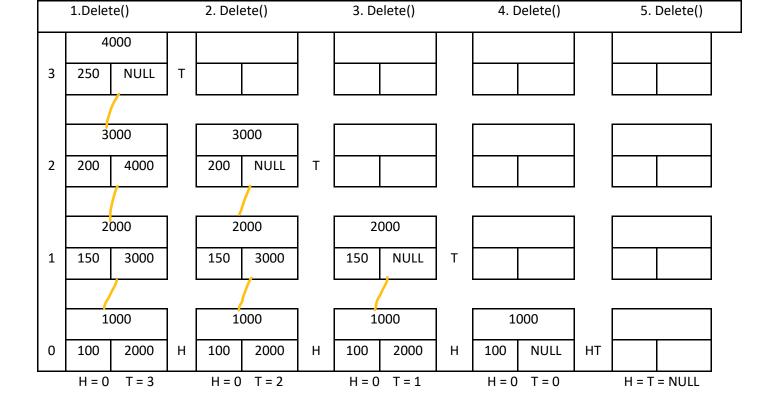
<u>Delete :-</u> If nodes are there we print the data of res and and apply free function to it. free function is used to remove memory of that data

	At First Node	General Condition
Empty LL		copy head to temp i.e;
Empty Lt	If Head and Tail are equal (i.e;	temp=head. Using while loop for
When LL is empty	Both are at first node)	condition temp->next->next copy
1 2 1 1 1 1	Copy head to res . res = head	temp->next is copied to temp
return NULL	Then equate head and tail to NULL	temp = temp->next and the loop
	[head = NULL; tail = NULL]	continues. Then copy tail to res
	since the existing LL becomes	res=tail , NULL to temp->next
	empty and finally return res	temp->next = NULL and finally
		return res

In the below table these are the steps for deleting different nodes :-

- In 1,2 & 3 Steps, last most step is deleted from a linked list and it's link with it's previous one is deleted and it's previous nodes next address is made as NULL and tail is moved to it's previous nodes.
- 4. Here we have deleted 150 with address 2000. Since there is no any other next node for 100 and 100 is the single node in the linked list. It's next node address is taken as NULL and there are no any link to it.
- 5. Since Head and Tail are at 0,After deleting node of 100, head and tail will be at NULL which indicates that the linked list is empty.

Now if we try to delete the nodes it returns NULL and displays No nodes.



Display:-

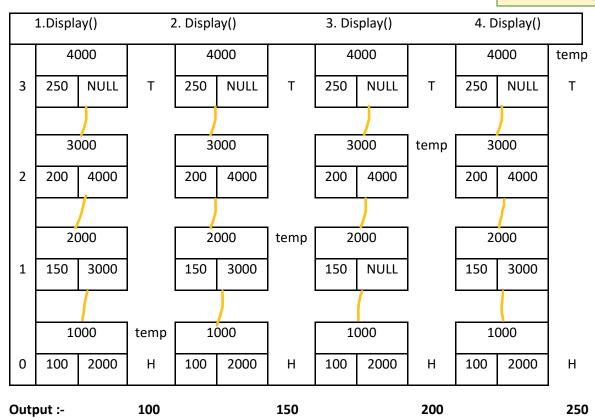
It displays nodes from head node to tail node by using the temp variable.

It displays the node which is at the position of temp.

When temp reaches tail node while loop stops Since tail->next == NULL.

If the linked list is empty, i.e; Head and tail are at NULL then it displays no nodes.

copy head to temp i.e; temp=head
Using while loop for condition temp
print temp and its data and next, then
copy temp->next is copied to temp
temp = temp->next and the loop
continues.



Single Linked List:-

Algorithms of insert at tail, delete at tail, insert at head, delete at tail, Insert by position, delete by position, reverse

and display.

Operations in SLL:

Insert at tail
Delete at tail
Insert at head
Delete at head
Insert by Position
Delete by Position
Display
reverse

Operations in SLL:

Insert at tail -> Same as insert in LL
Delete at tail -> Same as delete in LL
Display -> Same as display in LL

Insert at Tail :-

Empty LL

When LL is empty, copy NN to head and tail i.e; head = NN; tail = NN

General condition

Copy NN to tail->next i.e; tail->next = NN and copy NN to tail i.e; tail = NN

Display

copy head to temp i.e; **temp=head**Using while loop for condition **temp**print temp and its data & next,then
copy temp->next is copied to temp **temp = temp->next** and the loop
continues.

It displays nodes from head to tail in any case.

Since there is no any data in the node, Linked List is empty i.e; Head and tail are at NULL.

head = NULL tail = NULL

Insert(100): - Since this is the First node and there are no any next nodes to it. It's next node's address is taken as NULL.

1000 head = 0 tail = 0

Insert(200) :- Here we are inserting the 200 with address 2000 which will be linked to 100 Node by changing the next node of 100 as 2000 and take next node's address of 200 as NULL.

1000 2000 | 2000 | head = 0 tail = 1

Insert(300): Here we are inserting the 300 with address 3000 which will be linked to 200 Node by changing the next node of

 200 as 3000 and take next node's address of 300 as NULL

 1000
 2000
 3000
 head = 0 tail = 2

 100
 2000
 3000
 NULL

Insert(400): Here we are inserting the 400 with address 4000 which will be linked to 300 Node by changing the next node of 300 as 4000 and take next node's address of 400 as NULL

1000 2000 3000 4000 head = 0 tail = 3

<u>Delete at tail:</u> If nodes are there we print the data of res and and apply free function to it. free function is used to remove memory of that data.

Empty LL

When LL is empty return **NULL** to print **no nodes.**

At First Node

If Head and Tail are equal (i.e; Both are at first node)

Copy head to res . res = head

Then equate head and tail to NULL

[head = NULL; tail = NULL]

since the existing LL becomes empty and finally return res

General Condition

copy head to temp i.e; temp = head.

Using while loop for condition

temp->next->next temp->next is

copied to temp temp = temp->next

and the loop continues. Then copy tail

to res res = tail, NULL to temp->next

temp->next = NULL and copy temp to

tail i.e; tail = temp and finally return

res

Delete() :- It de	eletes the	lastmost no	de 400 and r	emoves	s its link w	rith 300 l	ov chang	ing the ne	ext node's addre	ess of 300 as NULL.
1000		2000			3000			000	head = 0	
100 2000		<u> </u>	000	300			400	NULL	tail = 3	
Delete():- It de	eletes the	lastmost no	de 300 and r	emoves	s its link w	ith 200 l	y chang	ing the ne	ext node's addre	ess of 200 as NULL.
1000		2000		3	3000		d = 0			
100 2000		200 30	000	300	NULL	tai	1 = 2			
Delete() :- It de	eletes the	lastmost no	de 200 and r	emoves	s its link w	ith 100 l	y chang	ing the ne	ext node's addre	ess of 100 as NULL
1000		2000		ad = 0						
100 2000		200 NI	JLL ta	il = 1						
Delete():- It de	eletes the	single node	100 and cha	nges H	ead and T	ail to NU	JLL			
100	$ \begin{array}{c c} 100 & head = 0 \\ tail = 0 \end{array} $									
100 NULL	100 NULL tail = 0									
Now if we try to	o do delet	te operation	it prints No	nodes.	Since LL i	is empty				
Insert at Head	<u>l :-</u>									
	Empty I	LL			G	eneral	conditio	on		
When LI	is empt	y, copy NN	J		Copy	/ head t	o NN ->	next i.e;		
to h	ead and	tail i.e;						copy NN		
	head = N				to h	ead i.e;	head =	= NN		
	tail = N	N								
Since there is	head :	ata in the	node, Linke	d List	is empty i	i.e; Hea	d and ta	il are at l	NULL.	
` '	Since th	is is the Fi	rst node and	d there	are no an	ıy next ı	nodes to	it. It's n	ext node's add	dress is taken as
NULL.										
1000		d = 0 $d = 0$								
100 NULL										
Insert(200) :- node of 200 as		e are inser	ting the 200	with a	address 20	000 whi	ch will	be linked	to 100 Node	by changing the next
2000		1000	head :							
200 1000	10	00 NULL		- 1						
Insert(300) :- node of 300 as		e are inser	ting the 300	with a	address 30	000 whi	ch will	be linked	to 200 Node	by changing the next
3000		2000		1000		head = 0				
300 2000	20	00 1000	10		JLL	tail = 2				
						000 whi	ch will	be linked	to 300 Node	by changing the next
node of 400 as	s 3000 ar	nd take nex	t node's ad	dress o	$\frac{\text{of } 100}{1}$ as 1	NULL		7		
4000		3000		2000		1	000		head = 0	
400 3000		00 2000	20		000	100	NULL		tail = 3	
Delete at head			ere we prin	t the d	lata of res	s and ar	id apply	free fun	ction to it. fre	e function is used
to remove men	nory of t	nat data								

Empty LL

When LL is empty return **NULL**

At First Node

If Head and Tail are equal (i.e; Both are at first node).
Copy head to res . res = head
Then equate head and tail to NULL

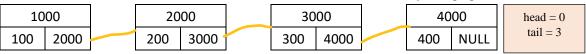
[head = NULL; tail = NULL] since the existing LL becomes empty and

finally return res

General Condition

copy head to temp i.e; temp=head.
Then copy head->next to head
head = head->next, NULL to temp>next temp->next = NULL and finally
return temp

Delete():- It deletes the frontmost node 100 and removes its link with 200 by changing the next node's address of 100 as NULL.



Delete():- It deletes the frontmost node 200 and removes its link with 300 by changing the next node's address of 200 as NULL.



Delete():- It deletes the frontmost node 300 and removes its link with 400 by changing the next node's address of 300 as NULL

3000		40	00	head = 0
300	4000	400	NULL	tail = 1

Delete():- It deletes the single node 400 and changes Head and Tail to NULL

40	00	head = 0
400	NULL	tail = 0

Now if we try to do delete operation it prints No nodes. Since LL is empty.

Reverse:- In this we have to declare some more pointer variables like cur, previous and next.

And Equate previous and next to NULL. [here cur=curnode = current node]

Empty LL

When LL is empty

Print No nodes

At First Node

If Head and Tail are equal (i.e; Both are at first node) Print No need since list contains single node.

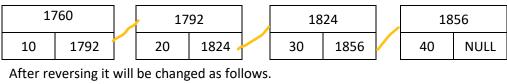
General Condition

copy head to tail and curnode i.e; tail = head, cur = head.

Then while loop for condition cur and copy cur->next to next next = cur->next, copy prev to cur->next cur->next = prev, equate cur to prev prev = cur and equate next to cur cur = next and loop terminates when cur becomes null.

Then copy prev to head head = prev

Suppose this is the Linked list before reversing.



18	56	1824		17	1792		17	60
40	1824	30	1792	20	1760		10	NULL

Insert by position :-

Empty LL

When LL is empty, copy NN to head and tail i.e;

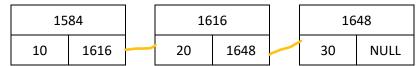
head = NN;

tail = NN

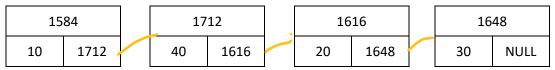
General Condition

Copy head to temp and run the loop between **p** = **1 to pos-1** by checking the condition **temp==NULL** if condition becomes **true print Insertion is not possible** and equate flag to 1 flag = 1 and break it and copy temp->next to temp temp = temp->next and terminate the loop. If flag == 0 copy temp->next to NN->next i.e; NN->next = temp-next and NN to temp->next i.e; Temp->next = NN.

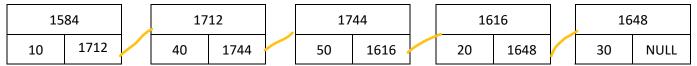
Suppose this is our Linked List



If we insert 40 at 1st position the changed linked list will be like this



If we insert 50 at 2nd position the changed linked list will be like this



If we try insert at position greater than (size -1)th position it prints **Insertion is not possible.**

Delete by position:-

Empty LL

When LL is empty

return **NULL**

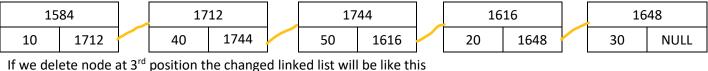
General Condition

Copy head to temp and run the loop between **p = 1 to pos-1** by checking the condition temp==NULL if condition becomes true return null and copy temp->next to temp **temp = temp->next** and terminate the loop.

Then copy temp->next to res i.e; res = temp->next,

Temp->next->next to temp->next i.e; **temp->next = temp->next->next** and equate NULL to res->next i.e; res->next = NULL and return res

Suppose this is our Linked List



15	84	1712		1744		1648	
10	1712	40	1744	50	1648	30	NULL

If we delete node at 1st position the changed linked list will be like this

15	84	17	44	1648	
10	1744	50	1648	30	NULL

If we try delete at position when the Linked list is empty it prints Deletion is not possible or No nodes.

Circular Linked List:-

Algorithms of insert at tail, delete at tail, insert at head, delete at tail, Insert by position, delete by position and

display.

Operations in CLL:

Insert at tail
Delete at tail
Insert at head
Delete at head
Insert by Position
Delete by Position
Display
reverse

Display

copy head to temp i.e; temp=head Using while loop for condition temp print temp and its data & next,then copy temp->next is copied to temp temp = temp->next and the loop continues.

It displays nodes from head to tail in any case.

Insert at Tail:-

Empty LL

When LL is empty, copy NN
to head and tail i.e;
head = NN;
tail = NN
copy head to head->next i.e;
head->next = head

General condition

Copy NN to tail->next i.e;

tail->next = NN and copy NN to
tail i.e; tail = NN and copy
head to tail->next i.e;
tail->next = head

Since there is no any data in the node, Linked List is empty i.e; Head and tail are at NULL.

head = NULL tail = NULL

Insert(100): - Since this is the First node and there are no any next nodes to it. It's next node's address is taken as it's address.

1000 100 1000

head = 0tail = 0

Insert(200) :- Here we are inserting the 200 with address 2000 which will be linked to 100 Node by changing the next node of 100 as 2000 and take next node's address of 200 as 1000.

1000 100 2000 2000 200 1000

head = 0tail = 1

Insert(300): Here we are inserting the 300 with address 3000 which will be linked to 200 Node by changing the next node of 200 as 3000 and take next node's address of 300 as 1000.

1000 100 2000 2000 200 3000 3000 300 1000

head = 0tail = 2

Insert(400): Here we are inserting the 400 with address 4000 which will be linked to 300 Node by changing the next node of 300 as 4000 and take next node's address of 400 as 1000

1000

2000 200 3000 300 4000

400 1000 degree 400 degree 400 degree 4000 degree 400

head = 0tail = 3

<u>Delete at tail :-</u> If nodes are there we print the data of res and apply free function to it. free function is used to remove memory of that data.

Empty LL

When LL is empty return **NULL** to print no nodes.

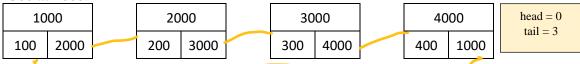
At First Node

If Head and Tail are equal (i.e; Both are at first node) Copy head to res i.e; res = head. Then equate head and tail to NULL [head = NULL; tail = NULL] since the existing LL becomes empty and finally return res

General Condition

copy head to temp i.e; temp=head. Using while loop for condition temp->next->next != head temp->next is copied to temp temp = temp->next and the loop continues. Then copy tail to res i.e; res = tail, head to temp->next i.e; temp->next = head, temp to tail i.e; tail = temp and NULL to res->next res->next = NULL and finally return res

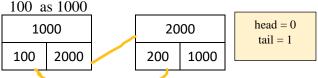
Delete():- It deletes the lastmost node 400 and removes its link with 300 by changing the next node's address of 300 as 1000.



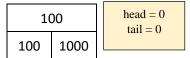
Delete():- It deletes the lastmost node 300 and removes its link with 200 by changing the next node's address of 200 as 1000.



Delete():- It deletes the lastmost node 200 and removes its link with 100 by changing the next node's address of



Delete():- It deletes the single node 100 and changes Head and Tail to NULL.



Now if we try to do delete operation it prints No nodes. Since LL is empty.

Insert at Head :-

Empty LL When LL is empty, copy NN to head and tail i.e; head = NN; tail = NN copy head to head->next i.e; head->next = head

General condition

Copy head to NN ->next i.e; NN -> next = head and copy NN to head i.e; **head = NN** and copy head to tail->next i.e; tail->next = head

Since there is no any data in the node, Linked List is empty i.e; Head and tail are at NULL.



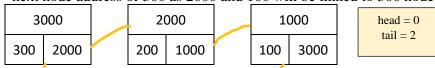
Insert(100): - Since this is the First node and there are no any next nodes to it. It's next node's address is taken as 1000.

10	000	head = 0
100	1000	tail = 0

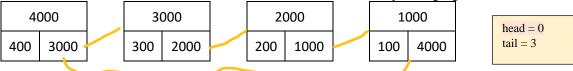
Insert(200):- Here we are inserting the 200 with address 2000 which will be linked to 100 Node by changing the next node address of 200 as 1000 and 100 will be linked to 200 node by changing next node address of 100 as 2000

20	2000		10	000	head = 0
200	1000		100	2000	tail = 1

Insert(300): Here we are inserting the 300 with address 3000 which will be linked to 200 Node by changing the next node address of 300 as 2000 and 100 will be linked to 300 node by changing next node address of 100 as 3000



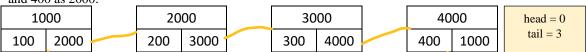
Insert(400): Here we are inserting the 400 with address 4000 which will be linked to 300 Node by changing the next node of 400 as 3000 and 100 will be linked to 400 node by changing next node address of 100 as 4000



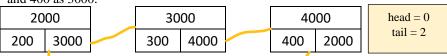
<u>Delete at head :-</u> If nodes are there we print the data of res and and apply free function to it. free function is used to remove memory of that data

At First Node If Head and Tail are equal **General Condition Empty LL** (i.e; Both are at first node). copy head to res i.e; res = head. Then copy Copy head to res . res = head head->next to head i.e; head = head->next, When LL is empty Then equate head and tail to NULL copy head to tail->next i.e; tail->next = head return **NULL** [head = NULL; tail = NULL] NULL to res->next i.e; res->next = NULL and since the existing LL becomes empty and finally finally return res. return res

Delete():- It deletes the frontmost node 100 and removes its link with 200 by changing the next node's address of 100 as NULL and 400 as 2000.



Delete():- It deletes the frontmost node 200 and removes its link with 300 by changing the next node's address of 200 as NULL and 400 as 3000.



Delete():- It deletes the frontmost node 300 and removes its link with 400 by changing the next node's address of 300 as NULL and 400 as 4000

and 40	U as 4000	J.			
3000			40	00	head = 0
300	4000		400	3000	tail = 1

Delete():- It deletes the single node 400 and changes Head and Tail to NULL

40	00	head = 0
400	4000	tail = 0

Now if we try to do delete operation it prints No nodes. Since LL is empty.

Insert by position :-

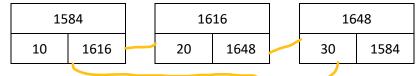
Empty LL

When LL is empty, copy NN to
head and tail i.e;
head = NN;
tail = NN
copy head to head->next i.e;
head->next = head

General Condition

Copy head to temp and run the loop between **p = 1 to pos-1** by checking the condition **temp->next == head** if condition becomes **true print Insertion is not possible** and return it otherwise copy temp->next to temp **temp = temp->next** and terminate the loop. Then copy temp->next to NN->next i.e; **NN->next = temp->next** and copy NN to temp -> next i.e; **temp->next = NN**

Suppose this is our Linked List



If we insert 40 at 1st position the changed linked list will be like this.

1584		1712		1616		1	648
10 17	712	40	1616	20	1648	30	1584

If we insert 50 at 2nd position the changed linked list will be like this

158	84	17	12	17	'44	16	16	16	48
10	1712	40	1744	50	1616	20	1648	30	1584

If we try insert at position greater than (size -1)th position it prints **Insertion is not possible.**

Delete by position:-

Empty LL

When LL is empty return **NULL**

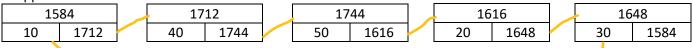
General Condition

Copy head to temp and run the loop between **p = 1 to pos-1** by checking the condition **temp->next == head** if condition becomes **true** print **'Deletion is not possible'** return **null** and copy temp->next to temp **temp = temp->next** and terminate the loop.

Then copy temp->next to res i.e; res = temp->next,

res->next to temp->next i.e; **temp->next = res->next** and equate NULL to res->next i.e; **res->next = NULL** and return **res**

Suppose this is our Linked List



If we delete node at 4th position the changed linked list will be like this

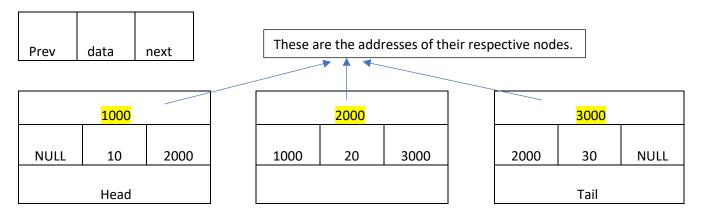
II WE de	iete noue	al +	position	the chang	cu III	ikeu iist w	WIII DE IIKE UII3					
15	84		17	12		17	44		16	48		
10	1712		40	1744		50	1648		30	1584		

If we delete node at 1st position the changed linked list will be like this

40 1744				
40 1744	50	1648	30	1712

Double Linked List:-

It consists of three nodes. 1^{st} node contains address of previous node, middle node contains data of that node and the last node contains address of next node



Algorithms of insert at tail, delete at tail, insert at head, delete at tail, Insert by position, delete by position, reverse and display.

Operations in SLL: Insert at tail

Insert at tail
Delete at tail
Insert at head
Delete at head
Insert by Position
Delete by Position
Display from head to tail
Display from tail to head

Insert at Tail:

Empty LL

When LL is empty, copy NN to head and tail i.e; head = NN; tail = NN

General condition

Copy NN to tail->next i.e;

tail->next = NN, copy tail to

NN-> prev i.e; NN->prev = tail

and copy NN to tail i.e;

tail = NN

Since there is no any data in the node, Linked List is empty i.e; Head and tail are at NULL.

Insert(100): - Since this is the First node and there are no any next nodes to it. It's next and previous node's address is taken as NULL.

	1000	
NULL	100	NULL

Insert(200) :- Here we are inserting the 200 with address 2000 which will be linked to 100 Node by changing the next node of 100 as 2000 and take previous node's address of 200 as 1000.

			 p. 01.00		
	1000			2000	
NULL	100	2000	1000	200	NULL

Insert(300) :- Here we are inserting the 300 with address 3000 which will be linked to 200 Node by changing the next node of 200 as 3000 and take previous node's address of 300 as 2000.

	1000		2000				3000	
NULL	100	2000	1000	200	3000	2000	300	NULL

<u>Delete at tail :-</u> If nodes are there we print the data of res and and apply free function to it. free function is used to remove memory of that data.

Empty LL

When LL is empty return **NULL** to print **no nodes.**

At First Node

If Head and Tail are equal (i.e;
Both are at first node)
Copy head to res . res = head
Then equate head and tail to NULL
[head = NULL; tail = NULL]
since the existing LL becomes
empty and finally return res

General Condition

copy tail to res i.e; res=tail, tail->prev to tail i.e;
tail = tail->prev
copy NULL to tail->Next and res->prev i.e;
tail -> Next = NULL
res -> prev = NULL
and finally return res

Delete():- It deletes the lastmost node 300 and removes its link with 200 by changing the next node's address of 200 and previous nodes address of 300 as NULL.

	1000			2000			3000	
NULL	100	2000	1000	200	3000	2000	300	NULL

Delete():- It deletes the lastmost node 200 and removes its link with 100 by changing the next node's address of 100 and previous nodes address of 200 as NULL.

	1000			2000	
NULL	100	2000	1000	200	NULL

Delete():- It deletes the single node 100 and changes Head and Tail to NULL

	1000	
NULL	100	NULL

Now if we try to do delete operation it prints No nodes. Since LL is empty.

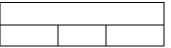
Insert at Head :-

Empty LL When LL is empty, copy NN to head and tail i.e; head = NN; tail = NN

General condition

Copy head to NN ->next i.e; NN > next = head, copy NN to head>prev i.e; head->prev = NN and
copy NN to head i.e;
head = NN

Since there is no any data in the node, Linked List is empty i.e; Head and tail are at NULL.



Insert(100): - Since this is the First node and there are no any next nodes to it. It's next and previous node's address is taken as NULL.

	1000	
NULL	100	NULL

Insert(200): Here we are inserting the 200 with address 2000 which will be linked to 100 Node by changing the next node of 200 as 1000 and take previous node's address of 100 as 2000.

	2000			1000	
NULL	200	1000	2000	100	NULL

Insert(300) :- Here we are inserting the 300 with address 3000 which will be linked to 200 Node by changing the next node of 300 as 2000 and take previous node's address of 200 as 3000.

3000				'	2000			1000	
NULL	300	2000		3000	200	1000	2000	100	NULL

Delete at head :- If nodes are there we print the data of res and and apply free function to it. free function is used to remove memory of that data

Empty LL

When LL is empty

return **NULL**

At First Node

General Conditioncopy head to res i.e; res = head. Then

head = head->next , NULL to head-

copy head->next to head i.e;

If Head and Tail are equal (i.e; Both are at first node).

Copy head to res . res = head

Then equate head and tail to NULL

[head = NULL; tail = NULL]

since the existing LL becomes empty and finally return **res**

head->prev = NULL

>prev and res->next i.e;

res->next = NULL and finally return res

Delete():- It deletes the frontmost node 100 and removes its link with 200 by changing the next node's address of 100 and previous nodes address of 200 as NULL.

1000								
NULL	100	2000						

2000									
1000	200	3000							

3000								
2000	300	NULL						

Delete():- It deletes the frontmost node 200 and removes its link with 300 by changing the next node's address of 200 and previous nodes address of 300 as NULL.

2000									
NULL	200	3000							

3000									
2000	300	NULL							

Delete():- It deletes the single node 300 and changes Head and Tail to NULL

	3000	
NULL	300	NULL

Now if we try to do delete operation it prints No nodes. Since LL is empty.

Insert by position :-

General Condition

Empty LL

When LL is empty, copy **NN** to head and tail i.e; head = **NN**;

tail = NN

Copy head to temp i.e; temp = head and run the loop between p = 1 to pos-1 by checking the condition temp==NULL if condition becomes true print Insertion is not possible and return it and copy temp->next to temp temp = temp->next and terminate the loop. Then copy temp->next to NN->next i.e; NN->next = temp->next, temp to NN->prev i.e; NN->prev = temp, also copy NN to temp->next i.e; temp->next = NN and NN to NN->next->prev i.e; NN->next->prev = NN.

Suppose this is our Linked List

	2584				
NULL	10	2616	2584	30	NULL

If we insert 40 at 1st position the changed linked list will be like this

	2584			2648		2616			
NULL	10	2648	2584	40	2616	2648	30	NULL	

If we insert 20 at 2nd position the changed linked list will be like this

2584			2648			2712				2616		
NULL	10	2648	2584	40	2712	2648	20	2616		2712	30	NULL

Delete by position:-

Empty LL

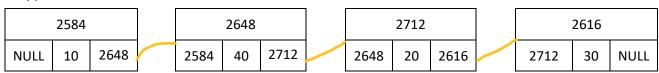
When LL is empty return **NULL**

General Condition

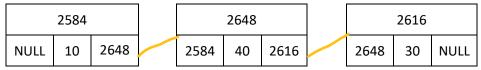
Copy head to temp and run the loop between **p = 1 to pos-1** by checking the condition **temp->next == head** if condition becomes **true** print **'Deletion is not possible' return null** and copy temp->next to temp **temp = temp->next** and terminate the loop.

Then copy temp->next to res i.e; res = temp->next ,Temp->next to temp->next i.e; temp->next = temp->next->next , copy temp to res->prev i.e; res->prev = temp and equate NULL to res->prev and res->next i.e; res->prev = NULL and res->next = NULL and return res

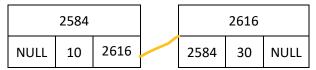
Suppose this is our Linked List



If we delete node at 2nd position the changed linked list will be like this



If we delete node at 1st position the changed linked list will be like this



Display from head to tail :-

Empty LL

When LL is empty it prints **No nodes**

copy head to temp i.e;

temp =head Using while loop for

condition temp print data of temp

then copy temp->next is copied to

temp temp = temp->next and the

loop continues.

		1.Disp	lay()		2.Di	splay()			3.0	isplay()			4.	Displ	ay()	
3		777			777				777					777		temp
	555	70	NULL	Т	555	70	NULL	Т	555	70	NULL	Т	555	70	NULL	Т
				_								_				
2		555				555				555		temp		555		
	333	50	777		333	50	777		333	50	777		333	50	777	
				_				-				_				
1		333				333		temp		333				333		
	111	30	555		111	30	555		111	30	555		111	30	555	
				_				•				•				
0		111		temp		111				111				111		
	NULL	10	333	Н	NULL	10	333	Н	NULL	10	333	Н	NULL	10	333	н
Output : - 10				_		30		•		50		•	70			

Display from tail to head :-

Empty LL

When LL is empty it prints **No nodes**

copy tail to temp i.e;

temp = tail Using while loop for

condition temp print data of temp
then copy temp->prev is copied to
temp temp = temp->prev and the
loop continues.

1.Display()					2.Dis	splay())		3.Display()					4.Display()			
3		777		temp		777				777				777			
	555	70	NULL	Т	555	70	NULL	Т	555	70	NULL	Т	555	70	NULL	Т	
				1				Ī	Г			1				Ī	
2		555	,			555	ı	temp		555	r	-		555			
	333	50	777		333	50	777		333	50	777		333	50	777		
				1	-			1	Г			7	Г			1	
1		333	,			333				333	T	temp		333			
	111	30	555		111	30	555		111	30	555		111	30	555		
				1	-			1	Г			1	Г			1	
0		111				111				111	T			111		temp	
	NULL	10	333	Н	NULL	10	333	Н	NULL	10	333	Н	NULL	10	333	Н	
	Output	t:-	70			50				30				10			

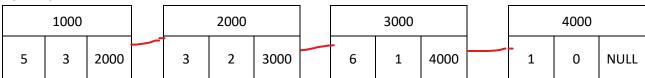
Polynomial Representation of Linked Lists:-

5x^3+9x^2+4x+10

	1000			2000			3000			4000	
coeff	pow	next	coeff	pow	next	coeff	pow	next .	coeff	pow	next
5	3	2000	9	2	3000	4	1	4000	10	1	NULL

For two polynomials:-

5x^3+3x^2+6x+1



 $4x^4 + 3x^3 + 2x^2 + 6x + 2$

	250			350			450		•	550			650	
4	4	350	 3	3	450	 2	2	550	6	1	650	2	0	NULL

Their resultant will be there sum

	2500		•	3500			4500			5500			6500	
4	4	3500	8	3	4500	5	2	5500	12	1	6500	3	0	NULL

Sparks Matrix :-

CR	0	1	2	3	4
0					
1					
2					
3					
4					

