**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.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data | Address 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.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 | |  | 2000 | |  | 3000 | |  | 4000 | |
| 10 | 2000 |  | 20 | 3000 |  | 30 | 4000 |  | 40 | NULL |
| Head | |  |  |  |  |  | |  | Tail | |

**Basic operations of Linked List:-:-**

Algorithms of Insert , Delete and Display

**Basic Operations in LL:**

Insert Delete Display

|  |
| --- |
| General terms in all algorithms:-  NN = NewNode  res = result or value |

**Insert :-**

**General condition**

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

**Empty LL**

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. | | | 2.Insert(100) | | | | | 3.Insert(150) | | | | 4.Insert(200) | | | | 5.Insert(250) | | | |
|  |  | | |  |  | |  | |  | |  | |  | |  | | 4000 | |  |
| 3 |  |  | |  |  |  |  | |  |  |  | |  |  |  | | 250 | NULL | T |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |
|  |  | | |  |  | |  | |  | |  | | 3000 | |  | | 3000 | |  |
| 2 |  |  | |  |  |  |  | |  |  |  | | 200 | NULL | T | | 200 | 4000 |  |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |
|  |  | | |  |  | |  | | 2000 | |  | | 2000 | |  | | 2000 | |  |
| 1 |  |  | |  |  |  |  | | 150 | NULL | T | | 150 | 3000 |  | | 150 | 3000 |  |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |
|  |  | | |  | 1000 | |  | | 1000 | |  | | 1000 | |  | | 1000 | |  |
| 0 |  |  | |  | 100 | NULL | HT | | 100 | 2000 | H | | 100 | 2000 | H | | 100 | 2000 | H |
|  | H = T = NULL | | |  | H = 0 T = 0 | |  | | H = 0 T = 1 | |  | | H = 0 T = 2 | |  | | H = 0 T = 3 | |  |

|  |  |  |
| --- | --- | --- |
| **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**. Using while loop for condition **temp->next->next** copy 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 finally return **res** |

**Delete :-** 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

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.

1. 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.
2. 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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.Delete() | | | 2. Delete() | | | | | 3. Delete() | | | | 4. Delete() | | | | 5. Delete() | | | | |
|  | 4000 | | |  |  | |  | |  | |  | |  | |  | |  | |  |
| 3 | 250 | NULL | | T |  |  |  | |  |  |  | |  |  |  | |  |  |  |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |
|  | 3000 | | |  | 3000 | |  | |  | |  | |  | |  | |  | |  |
| 2 | 200 | 4000 | |  | 200 | NULL | T | |  |  |  | |  |  |  | |  |  |  |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |
|  | 2000 | | |  | 2000 | |  | | 2000 | |  | |  | |  | |  | |  |
| 1 | 150 | 3000 | |  | 150 | 3000 |  | | 150 | NULL | T | |  |  |  | |  |  |  |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |  |  |  |
|  | 1000 | | |  | 1000 | |  | | 1000 | |  | | 1000 | |  | |  | |  |
| 0 | 100 | 2000 | | H | 100 | 2000 | H | | 100 | 2000 | H | | 100 | NULL | HT | |  |  |  |
|  | H = 0 T = 3 | | |  | H = 0 T = 2 | |  | | H = 0 T = 1 | |  | | H = 0 T = 0 | |  | | H = T = NULL | |  |

**Display :-**

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.

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.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.Display() | | | 2. Display() | | | | | 3. Display() | | | | 4. Display() | | | |
|  | 4000 | | |  | 4000 | |  | | 4000 | |  | | 4000 | | temp | |
| 3 | 250 | NULL | | T | 250 | NULL | T | | 250 | NULL | T | | 250 | NULL | T | |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |
|  | 3000 | | |  | 3000 | |  | | 3000 | | temp | | 3000 | |  | |
| 2 | 200 | 4000 | |  | 200 | 4000 |  | | 200 | 4000 |  | | 200 | 4000 |  | |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |
|  | 2000 | | |  | 2000 | | temp | | 2000 | |  | | 2000 | |  | |
| 1 | 150 | 3000 | |  | 150 | 3000 |  | | 150 | NULL |  | | 150 | 3000 |  | |
|  |  |  | |  |  |  |  | |  |  |  | |  |  |  | |
|  | 1000 | | | temp | 1000 | |  | | 1000 | |  | | 1000 | |  | |
| 0 | 100 | 2000 | | H | 100 | 2000 | H | | 100 | 2000 | H | | 100 | 2000 | H | |

**Output :- 100 150 200 250**

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

Display

**Operations in SLL:** 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.

**Insert at Tail :-**

**General condition**

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

**Empty LL**

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

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.  head = 0 tail = 0 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 next node’s address of 200 as NULL. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 2000 |  | 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 next node’s address of 300 as NULL | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 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  head = 0 tail = 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | |  | 4000 | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 4000 |  | 400 | NULL |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Delete at tail :-** 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 deletes the lastmost node 400 and removes its link with 300 by changing the next node’s address of 300 as NULL. | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | |  | 4000 | | head = 0 tail = 3 |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 4000 |  | 400 | NULL |  |  |  |  |  |  |  |  |  |  |
| Delete() :- It deletes the lastmost node 300 and removes its link with 200 by changing the next node’s address of 200 as NULL. | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | NULL |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Delete() :- It deletes the lastmost node 200 and removes its link with 100 by changing the next node’s address of 100 as NULL | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | NULL |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Delete() :- It deletes the single node 100 and changes Head and Tail to NULL | | | | | | | | | | | | | | | | | | | | |
| 100 | | head = 0 tail = 0 |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 100 | NULL |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Now if we try to do delete operation it prints No nodes. Since LL is empty. | | | | | | | | | | | | | | | | | | | | |

**Insert at Head :-**

**General condition**

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

**Empty LL**

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2000 | |  | 1000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 200 | 1000 |  | 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. | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3000 | |  | 2000 | |  | 1000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 300 | 2000 |  | 200 | 1000 |  | 100 | 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 400 as 3000 and take next node’s address of 100 as NULL | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4000 | |  | 3000 | |  | 2000 | |  | 1000 | |  | head = 0 tail = 3 | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 400 | 3000 |  | 300 | 2000 |  | 200 | 1000 |  | 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**  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. | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | |  | 4000 | | head = 0 tail = 3 |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 4000 |  | 400 | 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. | | | | | | | | | | | | | | | | | | | | |
| 2000 | |  | 3000 | |  | 4000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |
| 200 | 3000 |  | 300 | 4000 |  | 400 | 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 | |  | 4000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 300 | 4000 |  | 400 | NULL |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Delete() :- It deletes the single node 400 and changes Head and Tail to NULL | | | | | | | | | | | | | | | | | | | | |
| 4000 | | head = 0 tail = 0 |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 400 | NULL |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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.

|  |  |  |
| --- | --- | --- |
| **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** |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Suppose this is the Linked list before reversing. | | | | | | | | | | | | | | | | | | | | | | | |
| 1760 | | | |  | 1792 | | | |  | 1824 | | | |  | | 1856 | | | |  | |  |  |
| 10 | | 1792 | |  | 20 | | 1824 | |  | 30 | | 1856 | |  | | 40 | | NULL | |  | |  |  |
| After reversing it will be changed as follows. | | | | | | | | | | | | | | | | | | | | | | | |
| 1856 | | |  | | 1824 | | |  | | 1792 | | | |  | | 1760 | | | |
| 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 itandcopy 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 | | | | | | | | | | | | | |
| 1584 | |  | 1616 | |  | 1648 | |  |  |  |  |  |  |
| 10 | 1616 |  | 20 | 1648 |  | 30 | NULL |  |  |  |  |  |  |
| If we insert 40 at 1st position the changed linked list will be like this | | | | | | | | | | | | | |  |  |
| 1584 | |  | 1712 | |  | 1616 | |  | 1648 | |  |  |  |
| 10 | 1712 |  | 40 | 1616 |  | 20 | 1648 |  | 30 | NULL |  |  |  |
| If we insert 50 at 2nd position the changed linked list will be like this | | | | | | | | | | | | | |
| 1584 | |  | 1712 | |  | 1744 | |  | 1616 | |  | 1648 | |
| 10 | 1712 |  | 40 | 1744 |  | 50 | 1616 |  | 20 | 1648 |  | 30 | NULL |
| If we try insert at position greater than (size – 1)th position it prints **Insertion is not possible.** | | | | | | | | | | | | | |

|  |  |
| --- | --- |
| **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->nexti.e; **res->next = NULL** and return **res** |

**Delete by position :-**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Suppose this is our Linked List | | | | | | | | | | | | | |
| 1584 | |  | 1712 | |  | 1744 | |  | 1616 | |  | 1648 | |
| 10 | 1712 |  | 40 | 1744 |  | 50 | 1616 |  | 20 | 1648 |  | 30 | NULL |
| If we delete node at 3rd position the changed linked list will be like this | | | | | | | | | | | | | |
| 1584 | |  | 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 | | | | | | | | | | | | | |
| 1584 | |  | 1744 | |  | 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.

**Display**

**Operations in CLL:** Insert at tail Delete at tail Insert at head Delete at head Insert by Position Delete by Position Display reverse

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 :-**

**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**

**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**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | | head = 0 tail = 0 |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 | |  | 2000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 | |  | 2000 | |  | 3000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 | |  | 3000 | |  | 4000 | | head = 0 tail = 3 |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 4000 |  | 400 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Delete at tail :-** 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 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. | | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | |  | 4000 | | head = 0 tail = 3 |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 4000 |  | 400 | 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. | | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Delete() :- It deletes the lastmost node 200 and removes its link with 100 by changing the next node’s address of 100 as 1000 | | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Delete() :- It deletes the single node 100 and changes Head and Tail to NULL. | | | | | | | | | | | | | | | | | | | | | |
| 100 | | head = 0 tail = 0 |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 100 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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. | | | | | | | | | | | | | | | | | | | | | | | | | | |
|  | | 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 1000. | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1000 | | head = 0 tail = 0 |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 100 | 1000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2000 | |  | 1000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 200 | 1000 |  | 100 | 2000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3000 | |  | 2000 | |  | 1000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 300 | 2000 |  | 200 | 1000 |  | 100 | 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 | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4000 | |  | 3000 | |  | 2000 | |  | 1000 | |  | head = 0 tail = 3 | |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 400 | 3000 |  | 300 | 2000 |  | 200 | 1000 |  | 100 | 4000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |



|  |  |  |
| --- | --- | --- |
| **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 res i.e; **res = head**. Then copy head->next to head i.e; **head = head->next** , copy head to tail->next i.e; **tail->next = head** NULL to res->next i.e; **res->next = NULL** and finally return **res.** |

**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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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. | | | | | | | | | | | | | | | | | | | | |
| 1000 | |  | 2000 | |  | 3000 | |  | 4000 | | head = 0 tail = 3 |  | |  |  |  |  |  |  |  |
| 100 | 2000 |  | 200 | 3000 |  | 300 | 4000 |  | 400 | 1000 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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. | | | | | | | | | | | | | | | | | | | | |
| 2000 | |  | 3000 | |  | 4000 | | head = 0 tail = 2 |  | |  |  | |  |  |  |  |  |  |  |
| 200 | 3000 |  | 300 | 4000 |  | 400 | 2000 |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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. | | | | | | | | | | | | | | | | | | | | |
| 3000 | |  | 4000 | | head = 0 tail = 1 |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 300 | 4000 |  | 400 | 3000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Delete() :- It deletes the single node 400 and changes Head and Tail to NULL | | | | | | | | | | | | | | | | | | | | |
| 4000 | | head = 0 tail = 0 |  | |  |  | |  |  | |  |  | |  |  |  |  |  |  |  |
| 400 | 4000 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 itotherwisecopy 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 | | | | | | | | | | | | | |
| 1584 | |  | 1616 | |  | 1648 | |  |  |  |  |  |  |
| 10 | 1616 |  | 20 | 1648 |  | 30 | 1584 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| If we insert 40 at 1st position the changed linked list will be like this. | | | | | | | | | | | | | |
| 1584 | |  | 1712 | |  | 1616 | |  | 1648 | |  |  |  |
| 10 | 1712 |  | 40 | 1616 |  | 20 | 1648 |  | 30 | 1584 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| If we insert 50 at 2nd position the changed linked list will be like this | | | | | | | | | | | | | |
| 1584 | |  | 1712 | |  | 1744 | |  | 1616 | |  | 1648 | |
| 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->nexti.e; **res->next = NULL** and return **res** |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Suppose this is our Linked List | | | | | | | | | | | | | | |
| 1584 | |  | 1712 | |  | 1744 | |  | 1616 | |  | 1648 | |
| 10 | 1712 |  | 40 | 1744 |  | 50 | 1616 |  | 20 | 1648 |  | 30 | 1584 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| If we delete node at 4th position the changed linked list will be like this | | | | | | | | | | | | | | |
| 1584 | |  | 1712 | |  | 1744 | |  | 1648 | |  |  | |
| 10 | 1712 |  | 40 | 1744 |  | 50 | 1648 |  | 30 | 1584 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| If we delete node at 1st position the changed linked list will be like this | | | | | | | | | | | | | | |
| 1712 | |  | 1744 | |  | 1648 | |  |  | |  |  | |
| 40 | 1744 |  | 50 | 1648 |  | 30 | 1712 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Double Linked List:-**

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

These are the addresses of their respective nodes.

|  |  |  |
| --- | --- | --- |
| Prev | data | next |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 | | |  | 2000 | | |  | 3000 | | |
| NULL | 10 | 2000 |  | 1000 | 20 | 3000 |  | 2000 | 30 | NULL |
| Head | | |  |  | | |  | Tail | | |

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 from head to tail Display from tail to head

**Insert at Tail :-**

**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**

**Empty LL**

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

|  |  |  |
| --- | --- | --- |
| **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 at tail :-** 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.

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

**Insert at 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 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**  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 res i.e; **res = head**. Then copy head->next to head i.e; **head = head->next** , NULL to head->prev and res->next i.e; **head->prev = NULL 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 | | | |  | 2000 | | |  | 3000 | | |  |  |  |  | |  | |
| NULL | | 100 | 2000 |  | 1000 | 200 | 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 | | | |  | 3000 | | |  |  |  |  |  |  |  |  |  | |
| NULL | | 200 | 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 :-**

|  |  |
| --- | --- |
| **Empty LL**  When LL is empty, copy **NN** to head and tail i.e; **head = NN; tail = NN** | **General Condition**  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 itandcopy 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 | | |  | 2616 | | |  |  |  |  |  |  |  |  |
| 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 |

|  |  |
| --- | --- |
| **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->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->nexti.e; **res->prev = NULL and res->next = NULL** and return **res** |

**Delete by position :-**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Suppose this is our Linked List | | | | | | | | | | | | | | |
| 2584 | | |  | 2648 | | |  | 2712 | | |  | 2616 | | |
| NULL | 10 | 2648 |  | 2584 | 40 | 2712 |  | 2648 | 20 | 2616 |  | 2712 | 30 | NULL |
| If we delete node at 2nd position the changed linked list will be like this | | | | | | | | | | | | | | |
| 2584 | | |  | 2648 | | |  | 2616 | | |  |  |  |  |
| NULL | 10 | 2648 |  | 2584 | 40 | 2616 |  | 2648 | 30 | NULL |  |  |  |  |
| If we delete node at 1st position the changed linked list will be like this | | | | | | | | | | | | | | |
| 2584 | | |  | 2616 | | |  |  |  |  |  |  |  |  |
| NULL | 10 | 2616 |  | 2584 | 30 | NULL |  |  |  | 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. |  |  |  |  |

**Empty LL**

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

**Display from head to tail :-**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.Display() 2.Display() 3.Display() 4.Display() | | | | | | | | | | | | | | | | | |
| 3 | 777 | | |  | 777 | | |  | 777 | | |  | 777 | | | temp | |
|  | 555 | 70 | NULL | T | 555 | 70 | NULL | T | 555 | 70 | NULL | T | 555 | 70 | NULL | T |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 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 | H | NULL | 10 | 333 | H | NULL | 10 | 333 | H | NULL | 10 | 333 | H |

**Output : - 10 30 50 70**

**Display from tail to head :-**

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.

**Empty LL**

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1.Display() 2.Display() 3.Display() 4.Display() | | | | | | | | | | | | | | | | | |
| 3 | 777 | | | temp | 777 | | |  | 777 | | |  | 777 | | |  | |
|  | 555 | 70 | NULL | T | 555 | 70 | NULL | T | 555 | 70 | NULL | T | 555 | 70 | NULL | T |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 2 | 555 | | |  | 555 | | | temp | 555 | | |  | 555 | | |  | |
|  | 333 | 50 | 777 |  | 333 | 50 | 777 |  | 333 | 50 | 777 |  | 333 | 50 | 777 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 1 | 333 | | |  | 333 | | |  | 333 | | | temp | 333 | | |  | |
|  | 111 | 30 | 555 |  | 111 | 30 | 555 |  | 111 | 30 | 555 |  | 111 | 30 | 555 |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 111 | | |  | 111 | | |  | 111 | | |  | 111 | | | temp | |
|  | NULL | 10 | 333 | H | NULL | 10 | 333 | H | NULL | 10 | 333 | H | NULL | 10 | 333 | H |

**Output : - 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 :-**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 | | |  | 2000 | | |  | 3000 | | |  | 4000 | | |
| 5 | 3 | 2000 |  | 3 | 2 | 3000 |  | 6 | 1 | 4000 |  | 1 | 0 | NULL |

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

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

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

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

Their resultant will be there sum

**Sparks Matrix :-**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C R | 0 | 1 | 2 | 3 | 4 |
| 0 |  |  |  |  |  |
| 1 |  |  |  |  |  |
| 2 |  |  |  |  |  |
| 3 |  |  |  |  |  |  |
| 4 |  |  |  |  |  |

To insert data into matrix

|  |
| --- |
|  |