**Linked List**

Linked list is a linear collection of data element called node. Each node is divided into two field, the first part contains the information of the element and the second part contain the address field which stores the address of the next node in the list.

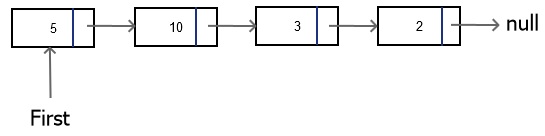


Fig: Linked list

**Singly Linear Linked List**

In this linked list, each node is divided into two parts information field and address field. In this list, there is sequential traversing from head to end of the list.

**Algorithm for operation in singly linear linked list**

* **Insertion at the beginning**

Insert\_front(head,item)

1. Start
2. Allocate a memory for a new node (n) and set n->data=item
3. If node already exists then display node exists and exit
4. Else if head is NULL then head=n
5. Else set n->next=head and head=n
6. Exit

* **Insertion at the end**

Insert\_back(head,item)

1. Start
2. Allocate a memory for a new node (n) and set n->data=item,

set n->next=null

1. If node already exists then display node exists and exit
2. Else if head is NULL then head=n
3. Else set ptr=head and traverse the ptr until ptr->next != NULL then set ptr->next=n
4. Exit

* **Insertion after a specific node**

Insert\_after(head,data\_to\_insert,item2)

1. Start
2. Allocate a memory for a new node (n) and set n->data=item,

set n->next=null

1. If node with item 2 data doesn’t exist then display node with the data doesn’t exist and exit
2. Else set ptr=head and traverse the ptr to the node with n->data=item2
3. Set n->next=ptr->next and set ptr->next=n
4. Exit

* **Insertion before a specific node**

Insert\_before(head,data\_to\_insert,item2)

1. Start
2. Allocate a memory for a new node (n) and set n->data=item,

set n->next=null

1. If node with item 2 data doesn’t exist then display node with the data doesn’t exist and exit
2. Else set ptr=head and traverse the ptr to the node with n->next->data=item2
3. Set n->next=ptr->next and set ptr->next=n
4. Exit

* **Deletion from the beginning**

Remove\_front(head)

1. Start
2. Set ptr=head
3. If head==NULL then display no node to remove and exit
4. Else if head->next=NULL then set head=NULL
5. Else set head=ptr->next
6. Delete ptr
7. Exit

* **Deletion from the end**

Remove\_back(head)

1. Start
2. If head==NULL then display no node to remove and exit
3. Else if head->next=NULL then set head=NULL
4. Else set ptr=head and preptr=head
5. Set ptr=ptr->next
6. Traverse the ptr and preptr until ptr->next!=NULL
7. Set preptr->next=NULL
8. Delete ptr
9. Exit

* **Deletion after a specific a node**

Remove\_after(head,item)

1. Start
2. If node with the data as item doesn’t exist then display node doesn’t exist and exit
3. Else set ptr=head and preptr=head
4. Set ptr=ptr->next
5. Traverse the preptr and ptr until preptr->data!=item2
6. If ptr is NULL then display no node to delete and exit
7. Else set preptr->next=ptr->next
8. Delete ptr
9. Exit