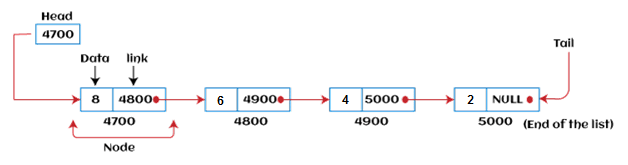
Linked list

**Linked list can be defined as the nodes that are randomly stored in the memory. A node in the linked list contains two parts, i.e., first is the data part and second is the address part. The last node of the list contains a pointer to the null. After array, linked list is the second most used data structure. In a linked list, every link contains a connection to another link**

Representation of a Linked list

Linked list can be represented as the connection of nodes in which each node points to the next node of the list. The representation of the linked list is shown below -



Types of Linked list

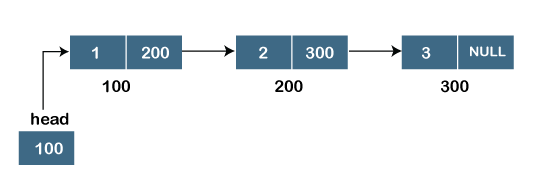
[Singly Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Singly)

[Doubly Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Doubly)

[Circular Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Circular)

[Doubly Circular Linked list](https://www.javatpoint.com/ds-types-of-linked-list#Doubly-Circular)

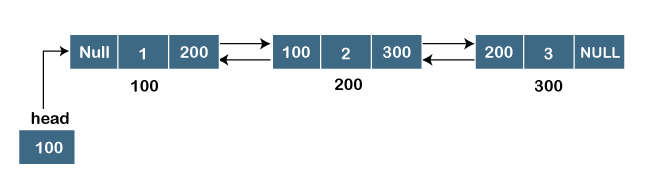
### Singly Linked list



**Representation of the node in a singly linked list**

1. **struct node**
2. **{**
3. **int data;**
4. **struct node \*next;**
5. **}**

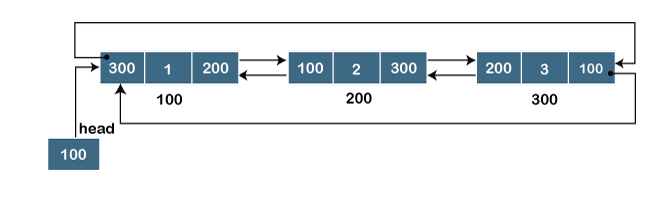
### Doubly linked list



Representation of the node in a doubly linked list

1. struct node
2. {
3. **int** data;
4. struct node \*next;
5. struct node \*prev;
6. }

### Circular linked list



Q.1 reverse a linked list

void reverse(){

     struct node\*pre,\*current,\*forward;

     pre=NULL;

     current=head;

     forward=head->next;

     while(forward!=NULL){

          current->next=pre;

          pre=current;

          current=forward;

          forward=forward->next;

     }

     current->next=pre;

     head=current;

}

Q.2 get length of linked list

int getlength(){

     struct node\*temp=head;

     int count=0;

     while(temp!=NULL){

          count++;

          temp=temp->next;

     }

     cout<<"length is :"<<count;

}

Q.3 get midd node nober in linked list

int getmidd(){

     int length= getlength();

     int midd= (length/2)+1;

     cout<< midd;

}

Q.4 get midd element

int getmidd1(){

     struct node\*temp = head;

     int length= getlength();

     int midd= (length/2);

     int count= 0;

     while( count < midd){

          temp = temp->next ;

          count++;

     }

     cout<< temp->data;

}

One more approach

struct ListNode\* middleNode(struct ListNode\* head) {

    if(head==NULL||head->next==NULL){

        return head;

    }

    if(head->next->next==NULL){

        return head->next;

    }

    struct ListNode\*slow=head;

    struct ListNode\*fast=head->next;

    while(fast!=NULL){

        fast=fast->next;

        if(fast!=NULL){

            fast=fast->next;

        }

        slow=slow->next;

    }

    return slow;

}