**WORKING WITH LINKED LIST**

**CREATION  
1.Singly-linked-list:**

**Creation**

Class Node{

Public:

Int data;

Node\* next;

Public :

Node(int data1 , Node\* next1)//for data with next node

{

Data=data1;

Next=next1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

}};

**Insertion**

Class Node{

Public:

Int data;

Node\* next;

Public :

Node(int data1 , Node\* next1)//for data with next node

{

Data=data1;

Next=next1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

}};

Node\* insert(Node\* node, int val){

Node\* temp= new Node(val,head);

Return temp;

}

**Deletion:**

Class Node{

Public:

Int data;

Node\* next;

Public :

Node(int data1 , Node\* next1)//for data with next node

{

Data=data1;

Next=next1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

}};

Node\* deleteNode(Node\* head){

Node\* temp =head;

While (temp->next->next!=nullptr){

Temp=temp->next;

}

Delete temp->next;

Temp->next=nullptr;

Return head;

**Delete at middle:**

Class Node{

Public:

Int data;

Node\* next;

Public :

Node(int data1 , Node\* next1)//for data with next node

{

Data=data1;

Next=next1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

}};

Node\* Delmiddle(Node\* head){

Node\* slow=head;

Node\* fast=head;

Fast=fast->next->next;

While(fast!=Null && fast->next!=Null){

Slow=slow->next;

Fast=fast->next->next;

}

Slow->next=slow->next->next;

Return head;

}

**2.Doubly-linked-list:**

**creation**

Class Node{

Public:

Int data;

Node\* next;

Node\* prev;

Public :

Node(int data1 , Node\* next1,Node\* prev1)//for data with next node

{

Data=data1;

Next=next1;

Prev=prev1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

Prev=nullptr;

}};

**Insertion**

Class Node{

Public:

Int data;

Node\* next;

Node\* prev;

Public :

Node(int data1 , Node\* next1,Node\* prev1)//for data with next node

{

Data=data1;

Next=next1;

Prev=prev1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

Prev=nullptr;

}};

Node\* insertAtbeg(Node\* head, int data){

Node\* curr=new Node(data);

If head==null{

Curr=head

Return;

}

Curr->next=head;

Head->prev=curr;

Head=curr;

}

**Deletion**

Class Node{

Public:

Int data;

Node\* next;

Node\* prev;

Public :

Node(int data1 , Node\* next1,Node\* prev1)//for data with next node

{

Data=data1;

Next=next1;

Prev=prev1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

Prev=nullptr;

}};

Node\* delatend(Node\* head){

If (head==Null) || (head->next==Null)

Return;

While (temp->next->next!=NULL){

Temp=temp->next;

}

Delete temp->next;

Temp->next=nullptr;

Return head;

}

**Delete at middle**

Class Node{

Public:

Int data;

Node\* next;

Node\* prev;

Public :

Node(int data1 , Node\* next1,Node\* prev1)//for data with next node

{

Data=data1;

Next=next1;

Prev=prev1;

}

Node(int data1)//if data alone given without next node

{

Data=data1;

Next=nullptr;

Prev=nullptr;

}};

Node \* delatmid((Node\* head){

If head==null || head->next==null

Return;

Node\* Slow=head;

Node\* Fast=head;

Fast=fast->next->next;

While(fast!=Null && fast->next!=Null){

Slow=slow->next;

Fast=fast->next->next;

}

Slow->next=fast->prev;

Fast->prev->prev=slow;

Return head;

}