DATA STRUCTURE

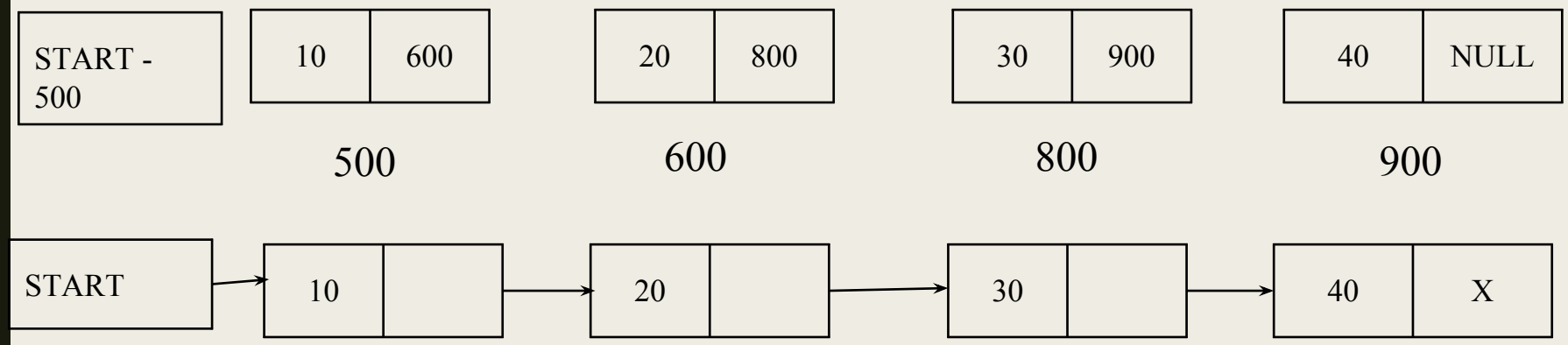
Sweta Suman

Linked List:

- To overcome the disadvantages of array, we use the concept of linked list
- Linked list is a linear data structure which stores data in a node
- Individual element is called a node which comprises of 2 parts:
  1. Data
  2. Next

- Data will store the information e.g: roll no, name, address etc
- Next is pointer which stores the address of next node

Memory Representation of Linked List:



START 500

Address	DATA	NEXT
500	10	600
600	20	800
700	-	-
800	30	900
900	40	NULL

Memory Allocation and Deallocation:

- **malloc:**

  syntax:

  ptr=(ptr return type *)malloc(size_to_be_allocated);

example:

```
int*a;
a=(int *) malloc( 10*sizeof(int));
a=(int*)malloc(10*4);
```

- **calloc:**

  syntax:

  ptr=(ptr return type *)calloc(n,element_size);

example:

```
int *a;
a=(int *) calloc( 10,sizeof(int));
a=(int*)calloc(10,4);
```

Memory Allocation and Deallocation:

- **realloc:**

  syntax:

  ptr=(int *)realloc(ptr,size);

  example:

  int *a;

  a=realloc(a,50*size0f(int));

- **free:**

  syntax:

  free(ptr);

  example:

  free(a);

Node Representation:


```
struct node
{
 int data;
 struct node *next;
};
struct node *start,*x,*y;


x= (struct node *)malloc(sizeof(struct node));
x->data=10;
y= (struct node *)malloc(sizeof(struct node));
y->data=20;
y->next=x;
x->next=NULL;
start=y;
```

Linked List:

Creating a node in a linked list:

```
struct node
{
int data;
struct node *next;
};
struct node *x,*y,*z;
void main()
{
int val;
Printf("enter value for 1st node");
```

Linked List:

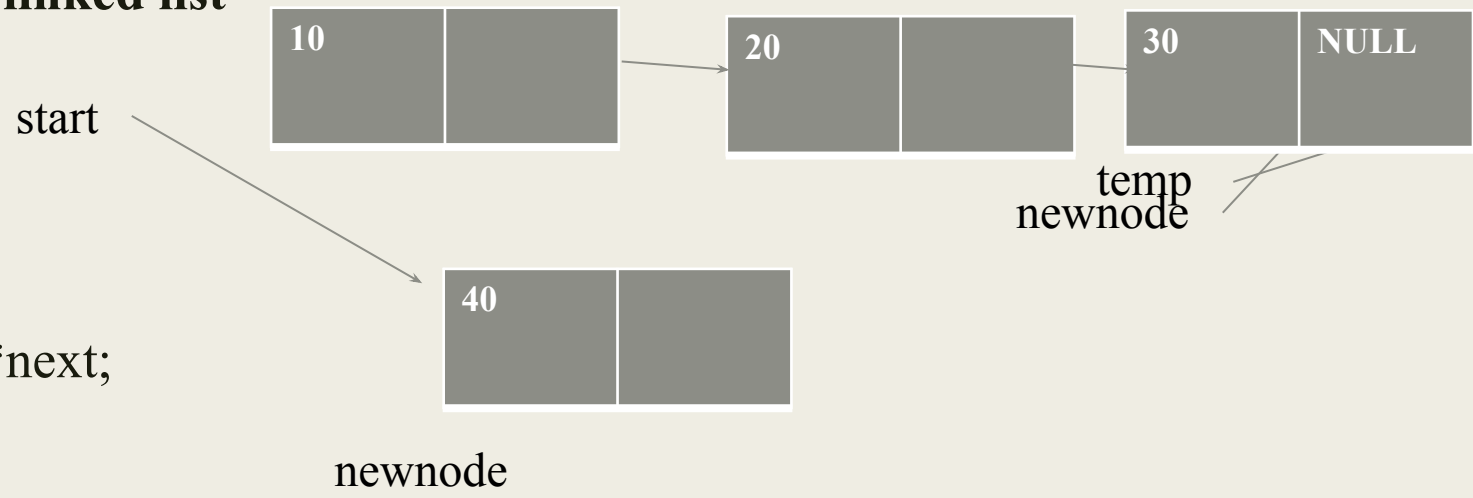
```
scanf("%d",&val);
x=(struct node*)malloc(sizeof(structnode));
x->data=val;
printf("Enter data for second node \n");
scanf(("%d",&val)
y=(struct node*)malloc(sizeof(structnode));
y->data=val;
printf("Enter data for third node \n");
scanf(("%d",&val)
z=(struct node*)malloc(sizeof(structnode));
z->data=val;
```

Linked List:

```
x->next=y;
y->next=z;
z->next=NULL;
printf("The value is node 1 is %d \n", x->data);
printf("The value is node 2 is %d \n", y->data);
printf("The value is node 3 is %d \n", z->data);
}
```

Linked List:

Creation of a linked list




```
struct node
{
   int data;
   struct node *next;
};
int main()
{
  struct node *start = NULL,*newnode, *temp;
   int val;
   printf("Enter a value for node");
   scanf("%d",&val);
```

Linked List:

```
do
   {
   newnode=(struct node *)malloc(sizeof(struct node));

   newnode->data=val;

 if(start==NULL)
{
       start=newnode;

       newnode->next=NULL;
  }

   else
   {
       temp=start;
       while(temp->next!=NULL)
       {
          temp=temp->next;
       }
```

Linked List:

```
temp->next=newnode;
        newnode->next=NULL;
    }
        printf("Enter a value or -1 to exit");
        scanf("%d",&val);
    }while(val!=-1);
temp=start;
    while(temp!=NULL)
    {
        printf("%d \t ",temp->data);
        temp=temp->next;
    }
  return 0;
}
```

Linked List:

Inserting a node in a linked list:


```
struct node *start;
struct node *newnode;
newnode=(struct node*)malloc(sizeof(struct node));
```

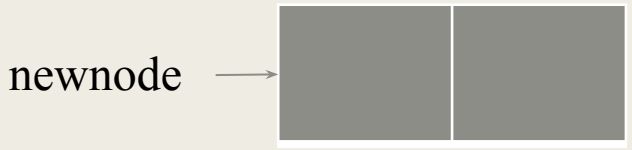
Case 1: Inserting as first node:

```
 if start==NULL \\ linked list empty
{
start=newnode;
newnode->next=NULL;
}
```

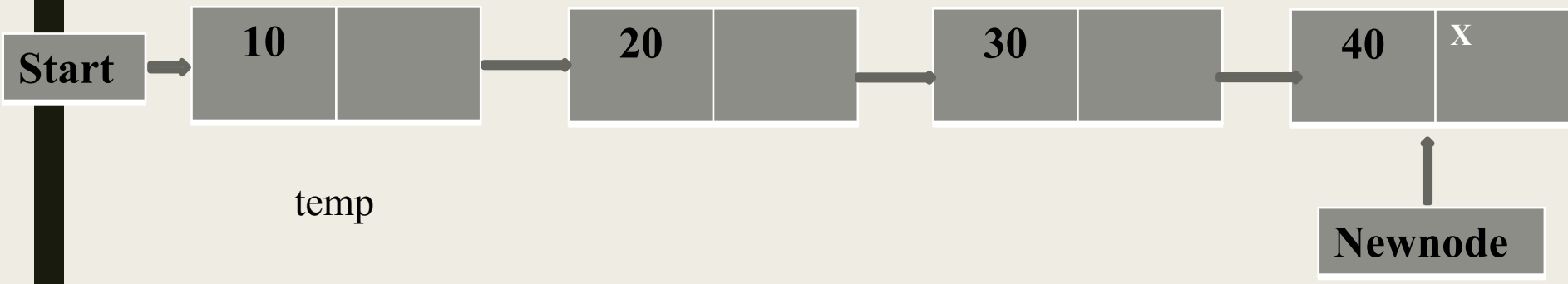
Linked List:

Case 2: Inserting as last node :

Traverse to the last element




```
struct node *temp;
temp=start;
While(temp->next!=NULL)
{
   temp=temp->next;
}
temp->next=newnode;
Newnode->next=NULL;
```

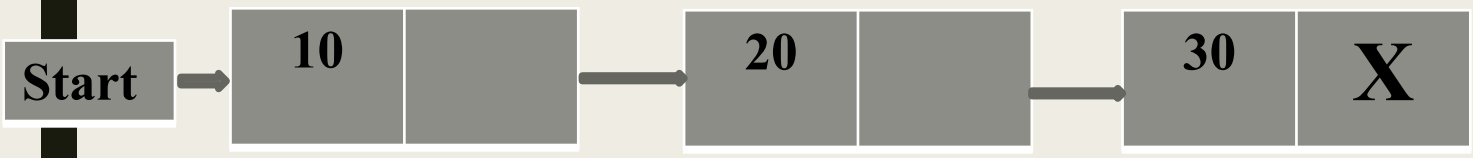

Start
10
20
30
40
X
temp
Newnode

Linked List:

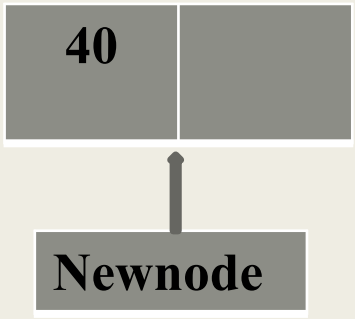
Case 3: Inserting an intermediate node :

Traverse to the element where newnode have to be inserted

```
printf("Enter the value after which insertion should be done");
scanf("%d",&val);
struct node *temp;
temp=start;
While(temp->data!=val)
{
   temp=temp->next;
}
newnode->next=temp->next;
temp->next=newnode;
```



temp



Linked List:

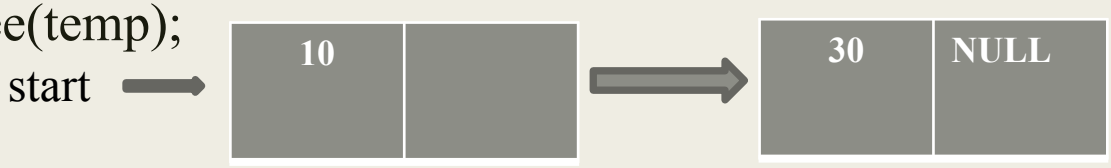
Deleting from a linked list:

Case 1: Deleting first node

temp=start;

start=temp->next;

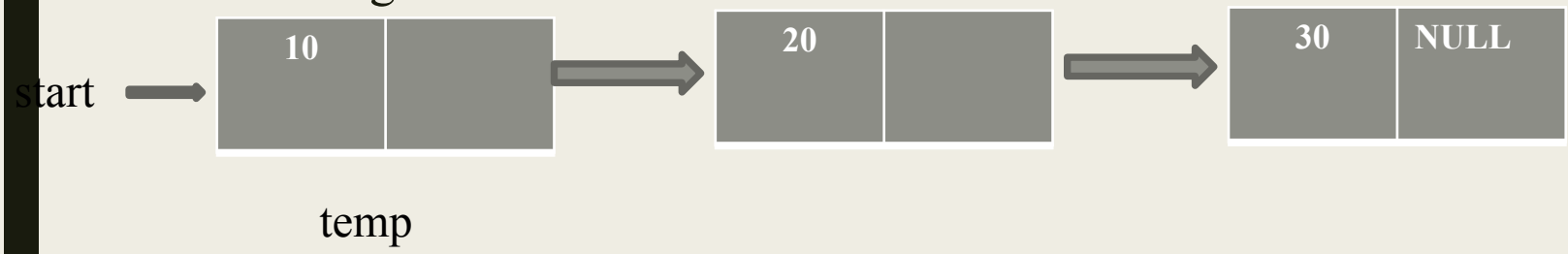
free(temp);



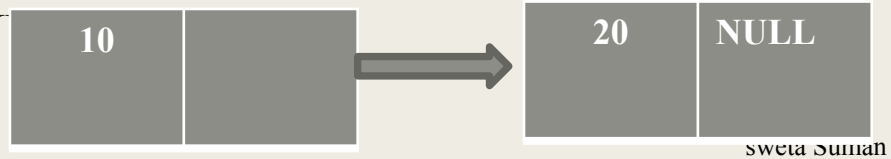
Linked List:

Deleting from a linked list:

Case 2: Deleting last node



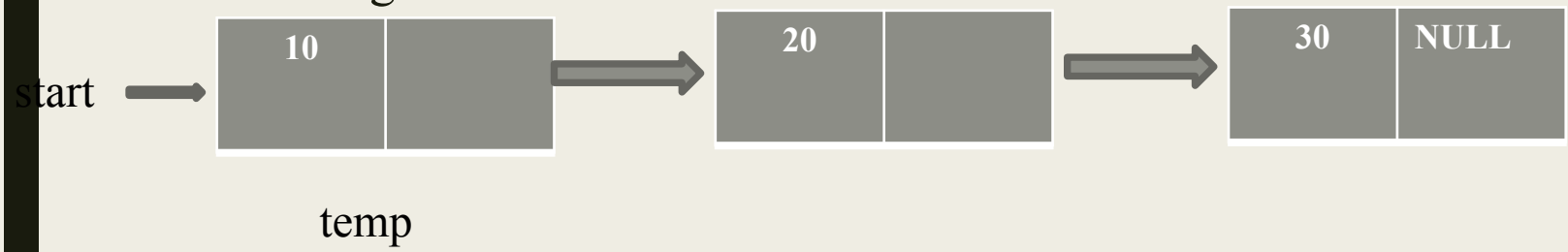

```
temp=start; pre=start;
while(temp->next!=NULL)
{pre=temp;
temp=temp->next;}
pre->next=N
free(temp);
```



Linked List:

Deleting from a linked list:

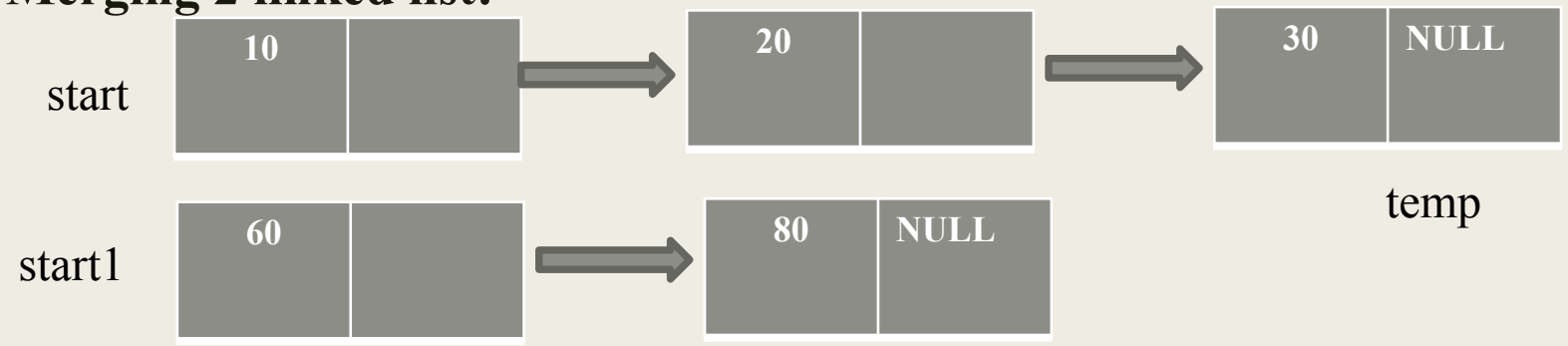
Case 3: Deleting an intermediate node



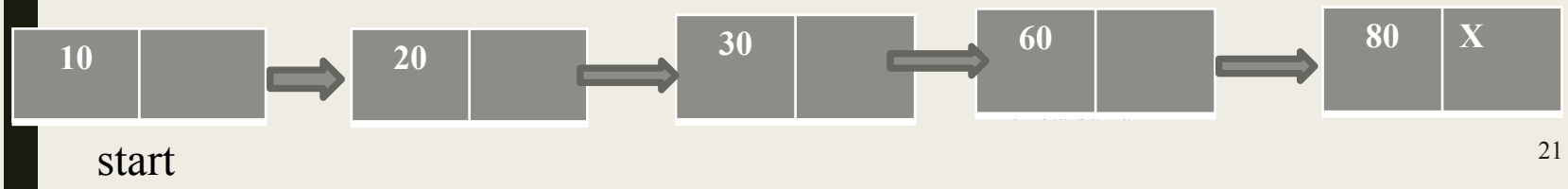

```
temp=start; pre=start;
printf("Enter the value to be deleted : ");
scanf("%d",&val);
while(temp->data!=Val)
{pre=temp;
temp=temp->next;}
pre->next=temp->next;
```

Linked List:

Merging 2 linked list:

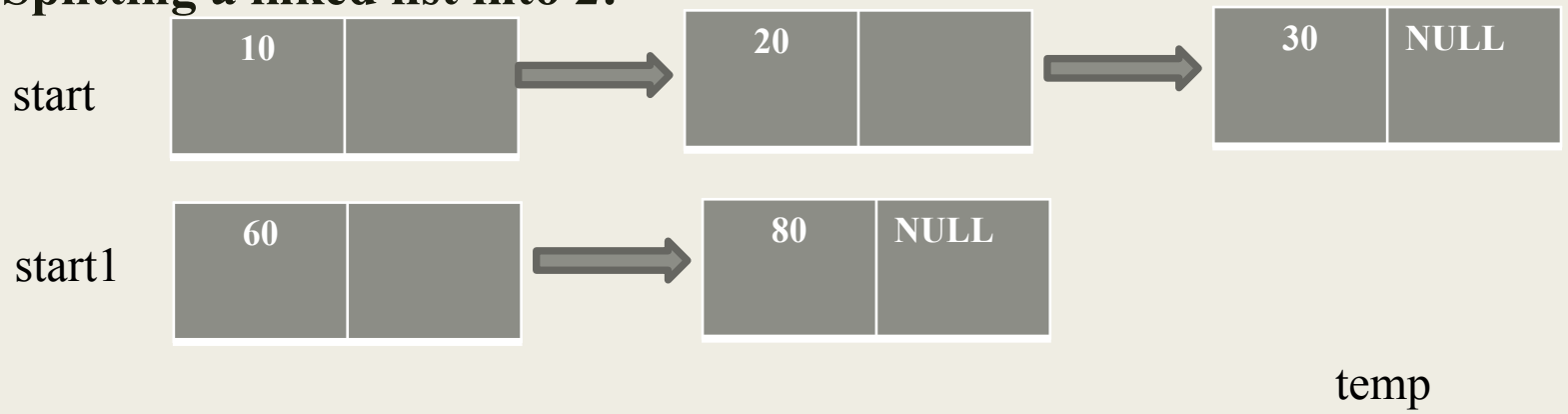



```
temp=start;
While(temp->next!=NULL)
{ temp=temp->next; }
temp->next=start1;
```



Linked List:

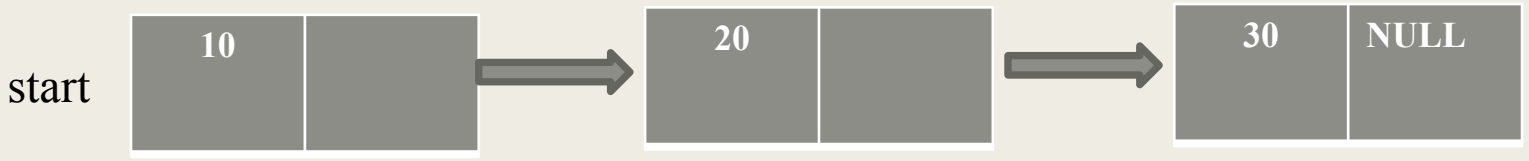
Splitting a inked list into 2:




```
temp=start;
While(temp->data!=val)
{ temp=temp->next; }
start1=temp->next;
temp->next=NULL;
```

Linked List:

copying a linked list into another:




```
temp=start;
while(temp!=NULL)
{
newnode=(struct node *)malloc(sizeof(struct node));
newnode->data=temp->data;
if(start1==NULL)
{
start1=newnode;
newnode->next=NULL; }
```

temp

Linked List:


```
else
{
temp1=start1;
while(temp1->next!=NULL)
{
temp1=temp1->next;
 }
temp1->next=newnode;
newnode->next=NULL
}
temp=temp->next;
```

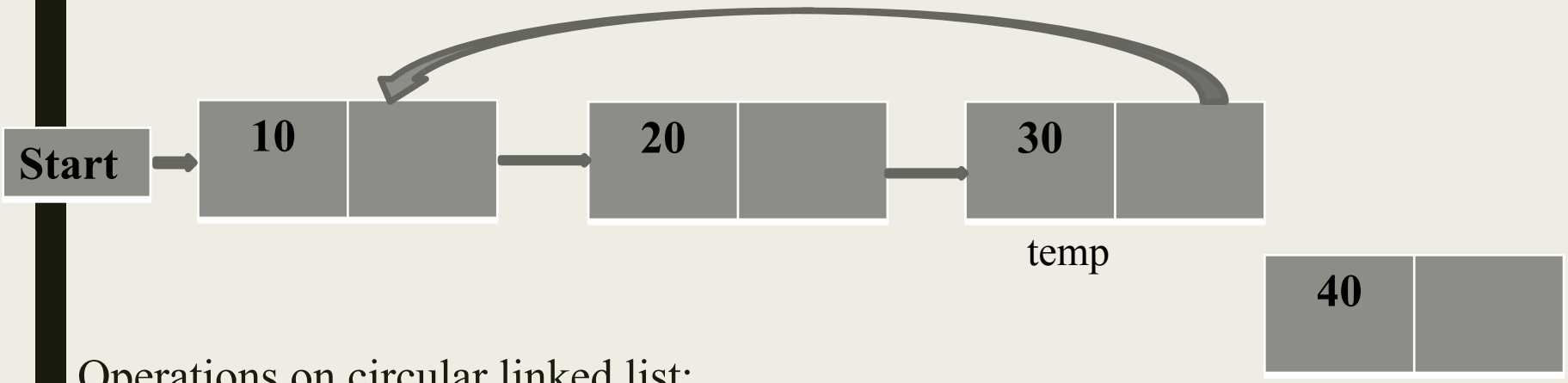
temp1

Reversing Linked List:

```
temp=start; pre=NULL;
while(temp!=NULL)
{
post=temp->next;
temp->next=pre;
pre=temp;
temp=post;
}
start=pre;
```

start

Circular Linked List:



Operations on circular linked list:

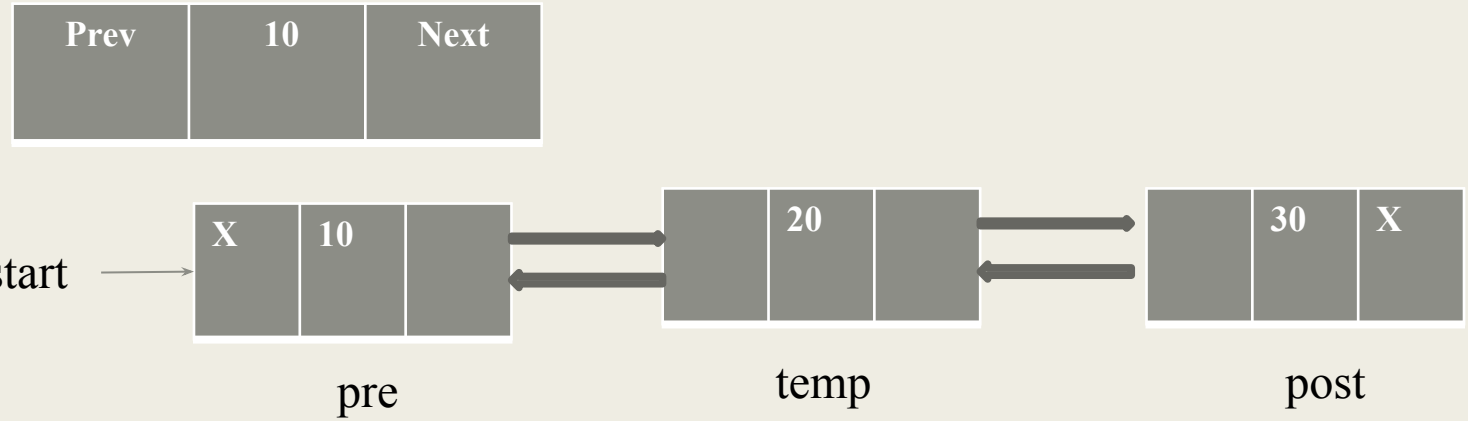
Inserting

Deleting

Doubly Linked List (2-way Linked list) :

A node in a doubly linked list comprises of 3 parts

Data, next pointer and previous pointer




```
struct node
{
int data;
struct node *next, *prev;
} ;
```

Operations performed on dobuly Linked list

Inserting:

Case 1: as first node

```
newnode=(struct node *)malloc(sizeof(struct node));
temp=start;
newnode->next=temp;
temp->prev=newnode;
start=newnode;
```

Case 2: as last node

```
temp=start;
while(temp->next!=NULL)
{
temp=temp->next; }
```

Operations performed on dobuly Linked list

```
emp->next=newnode;
ewnode->prev=temp;
ewnode->next=NULL;
```

ase 3: inserting as intermediate node

```
emp=start;
rintf("Enter value after which insertion should happen ");
canf("%d",&val);
hile(temp->data!=val)
 temp=temp->next;
ost=temp->next;

ewnode->next=post;
ost->prev=newnode;
```

Operations performed on dobuly Linked list

eleting :

ase 1: deleting first node

mp=start;

art=temp->next;

art->prev=NULL;

ee(temp);

ase 2: Deleting last node

mp=start;

hile(temp->next!=NULL)

re=temp;

mp=temp->next;

Operations performed on dobuly Linked list

eleting :

ase 3: deleting intermediate node

rintf("Enter the value to be deleted");

canf("%d",&val);

mp=start;

hile(temp->data!=val)

re=temp;

mp=temp->next;

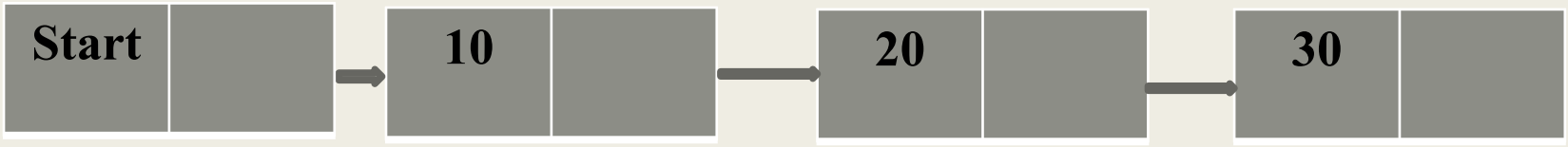
ost=temp->next;

re->next=post;

ost->prev=pre;

ee(temp);

Header Linked List:



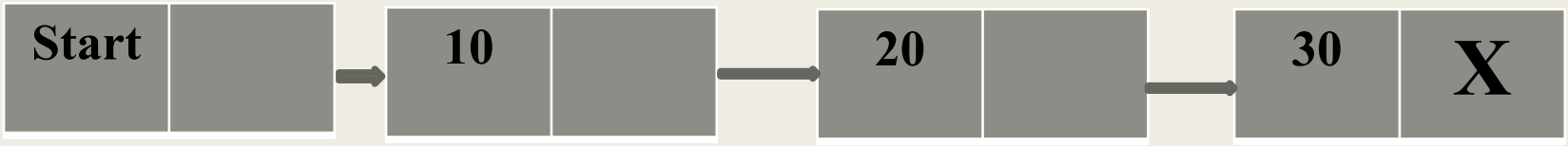
In Header linked list, we have a special node present at the beginning of the linked list.

In header linked list start does not refer to the first node but there is a header node which stored address of the first node

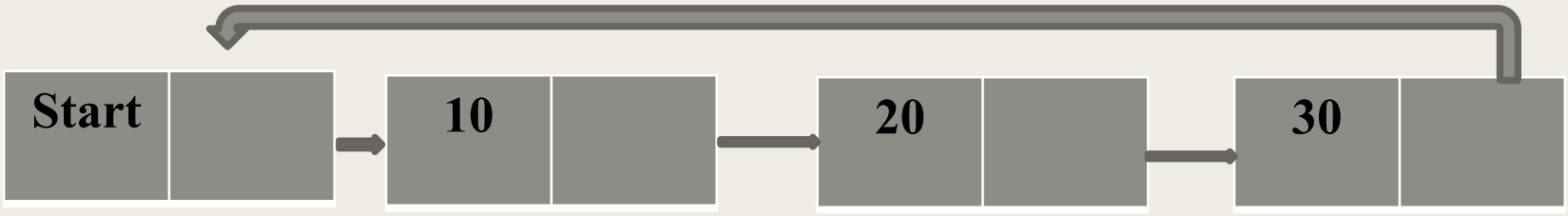
Types of Header linked list

Grounded Header Linked List

Circular Header Linked list



Grounded Header Linked List



Circular Header Linked List

Applications of Linked list:

Implementation of stacks and queues

Implementation of graphs : Adjacency list representation of graphs is most popular which is uses linked list to store adjacent vertices.

Dynamic memory allocation : We use linked list of free blocks.

Maintaining directory of names

Performing arithmetic operations on long integers

Manipulation of polynomials by storing constants in the node of linked list

representing sparse matrices

Operations performed on dobuly Linked list

eleting :

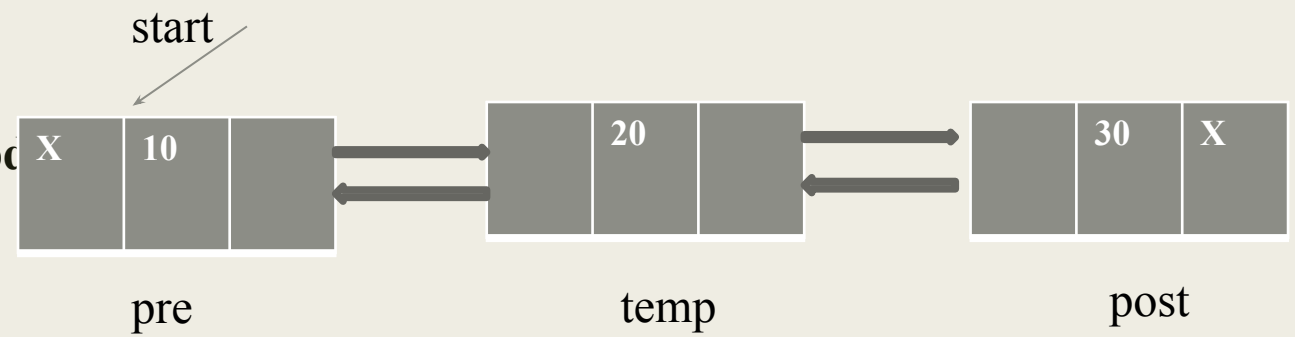
ase 1: deleting first noc

mp=start;

art=temp->next;

art->prev=NULL;

ee(temp);



ase 2: Deleting last node

mp=start;

hile(temp->next!=NULL)

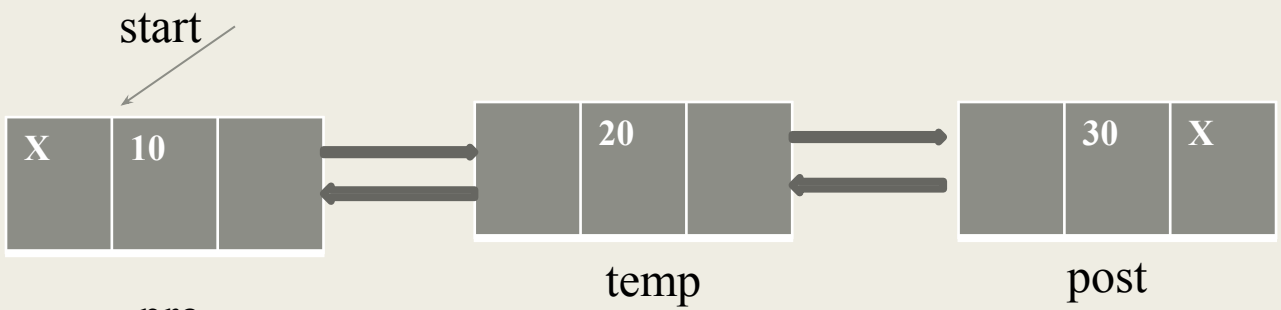
pre=t emp;

mp=temp->next; }

re->next=NULL;

ee(temp);

Operations performed on dobuly Linked list



ase 3:

eleting intermediate node

rintf(Enter the value to be deleted);

canf("%d",&val);

mp=start;

hile(temp->data!=val)

pre=temp;

mp=temp->next;

ost=temp->next; }

re->next=post;

ost->prev=pre;

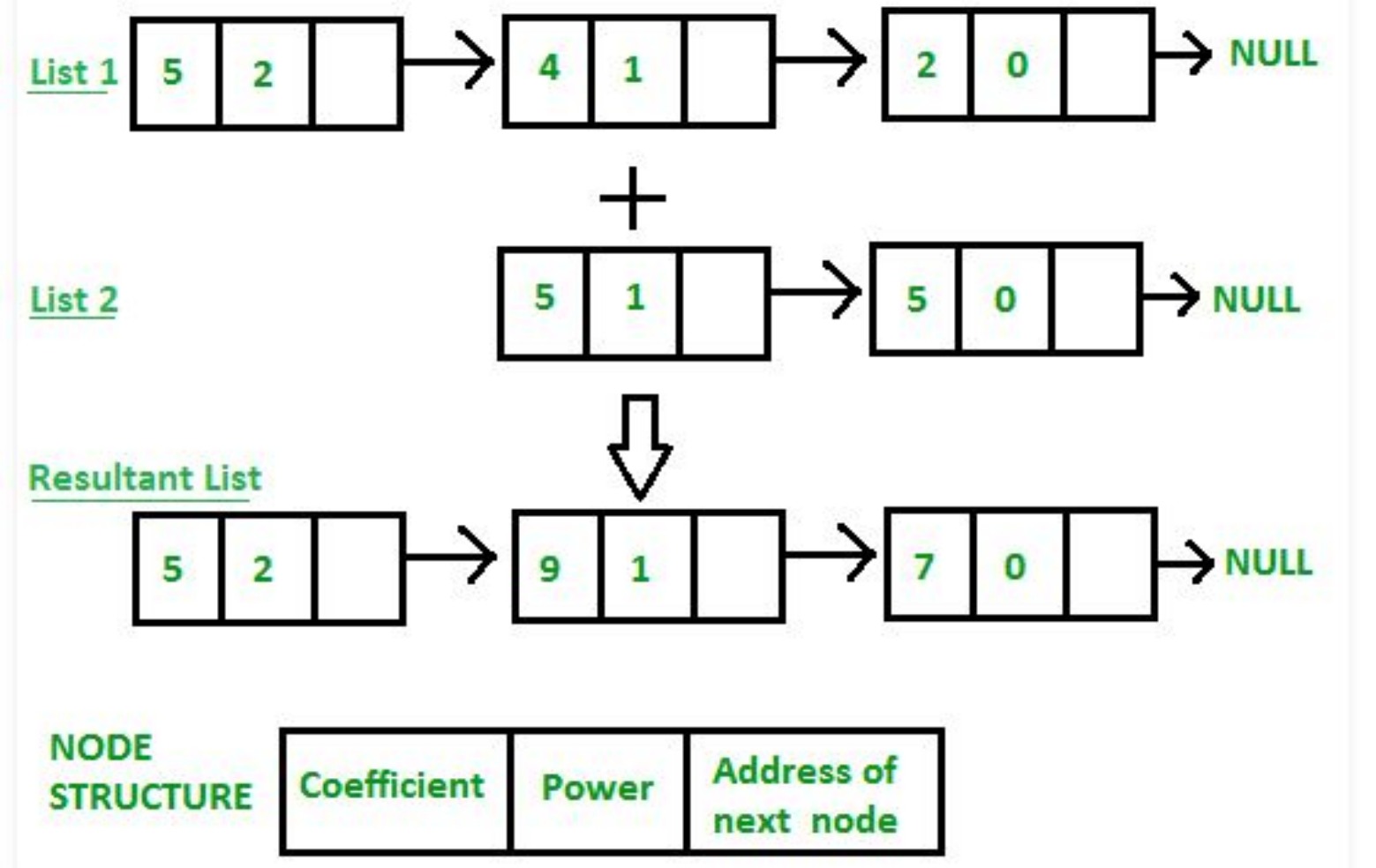
Polynomial Representation using Linked List

What is a Polynomial ?

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

```
struct node
{
int coeff;
int power;
struct node *x
};
```

Addition of 2 Polynomials:



5x2+4x+2 5x+5