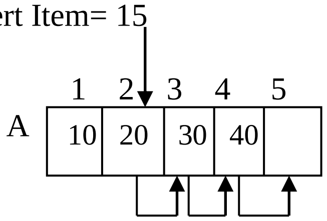
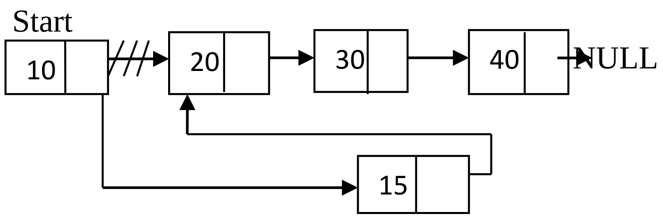
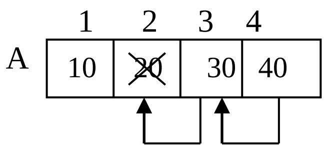
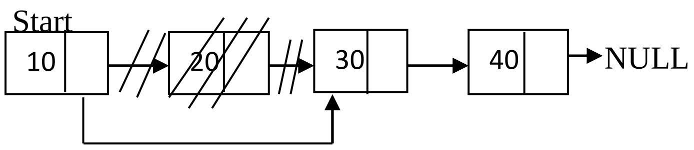
Linked lists

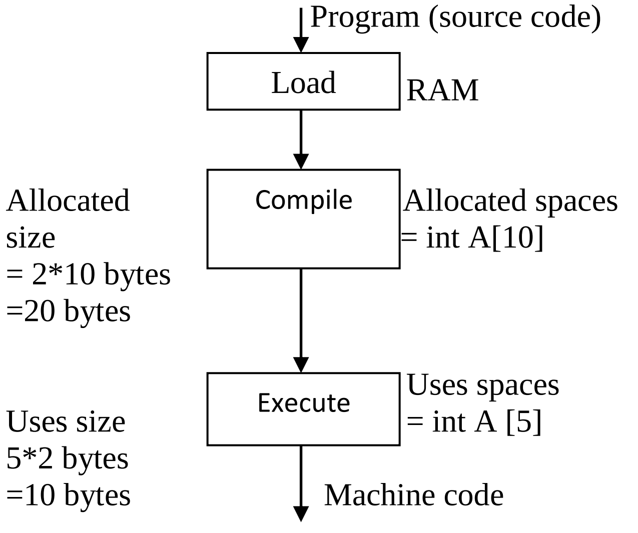
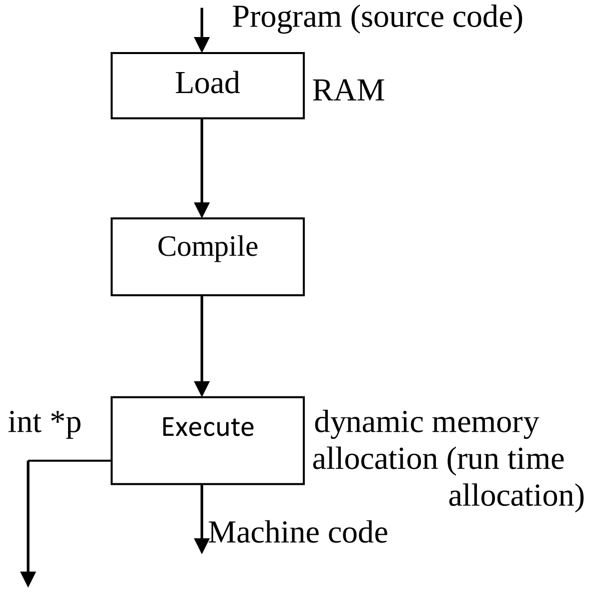
Dr. Ohidujjaman Tuhin
Assistant Professor
Dept. of CSE , UIU

<u>Introduction to linked lists</u>

A ***linked list*** is a linear data structure where each element is a separate object. Each element (we will call it a node) of a ***list*** is comprising of two items - the data and a reference to the next node. The last node has a reference to null.

3.2 MERITS AND DEMERITS OF LINKED LISTS OVER ARRAY

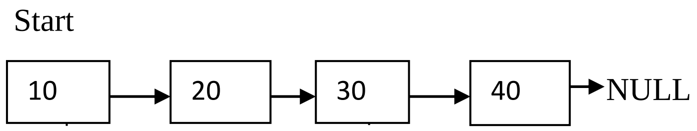
Array	Linked List
1) Insertion is Difficult because of movement of several elements and also takes more time (i.e time consuming). Insert Item= 15 	1) Insertion is easy because of no need movement of a single element. By creating additional node it is possible as shown in the following figure. Insert Item=15 
2) Deletion is difficult because of the similar reason of insertion as shown in the following figure. Delete Item= 20 	2) Deletion is easy because of deleting the node where the delete item is located. Delete Item=20 

3) Allow static memory allocation  Program (source code) Load RAM Compile Allocated size = 2*10 bytes =20 bytes Allocated spaces = int A[10] Execute Uses spaces = int A [5] Uses size 5*2 bytes =10 bytes Machine code Memory wastage= [Allocated memory size - Uses memory] = 20-10 bytes = 10 bytes	3) Allow dynamic memory allocation  Program (source code) Load RAM Compile int *p Execute dynamic memory allocation (run time allocation) Machine code Here starts to allocate memory and also use the allocated memory. So no wastage of memory.
4) Binary search is easy because of index available.	4) Binary search is difficult because of no index at all.
5) Location must be contiguous.	5)Not necessary to be contiguous
6)Homogeneous data structure	6) May be homogeneous or heterogeneous.

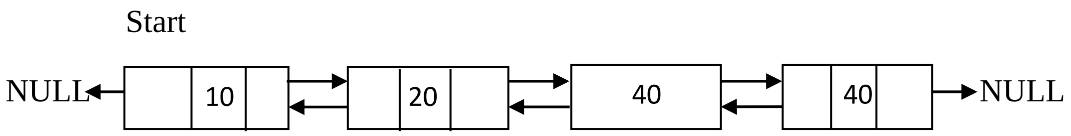
Different types of linked lists

There are several types of linked lists. They are depicted as below:

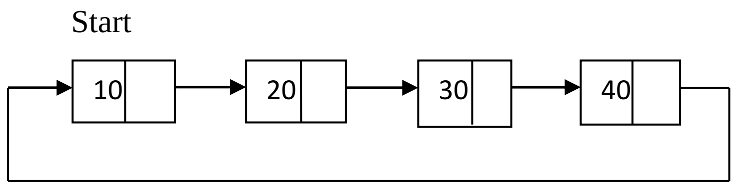
1) Liner or singular linked list



2) Doubly or two-way linked list

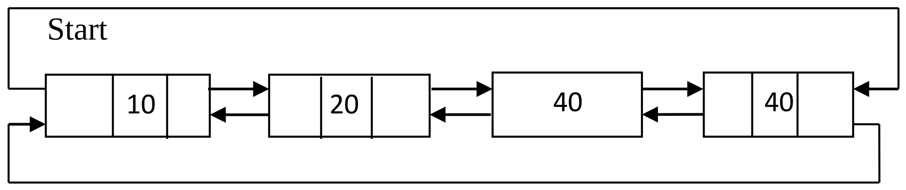


3) Linear circular linked list



No NULL pointer
(from last node to first node)

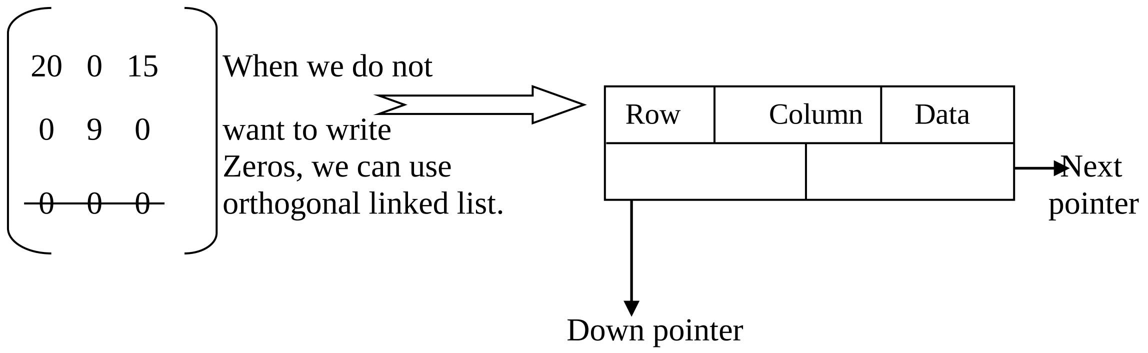
4) Two-way circular linked list



No NULL pointer

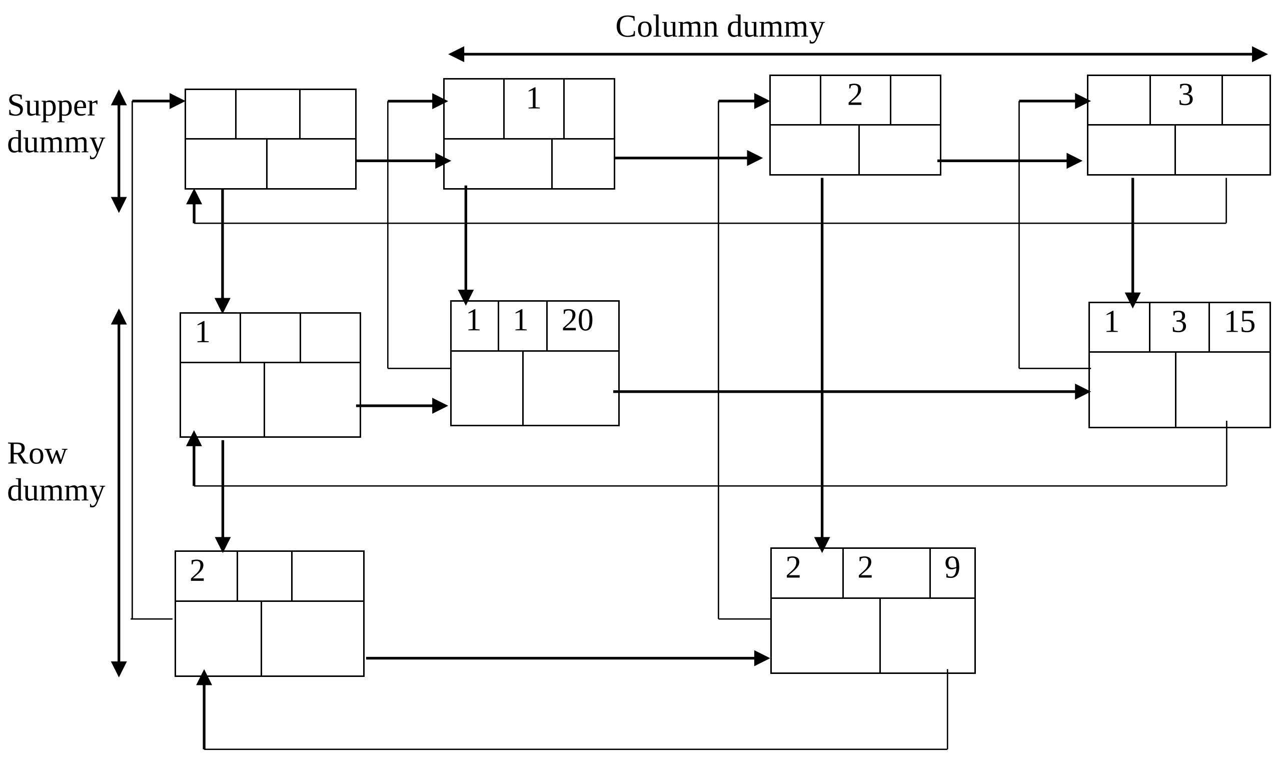
5) Orthogonal linked list

When there are more zeros than the data is called sparse matrix. Sparse matrix can be represented by orthogonal linked list. Orthogonal linked list basically is used to reduce the number of invalid data as well as save the memory.


20 0 15
0 9 0
~~0 0 0~~
When we do not want to write Zeros, we can use orthogonal linked list.
Row
Column
Data
Next pointer
Down pointer

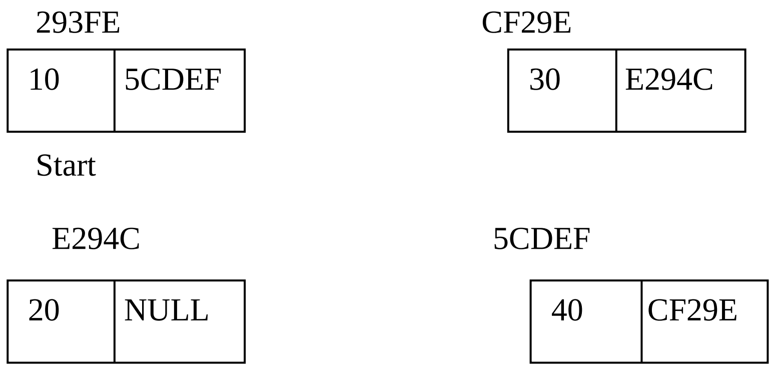
Number of valid data=3 (20, 15, 9)
Memory size = Number of valid data * Size of data type
= 3 * 2 (for integer data type)
= 6 bytes

Number of invalid data= 6
Memory size of invalid data= Number of invalid data * Size of data type
= 6 * 2 (for integer data type)
= 12 bytes

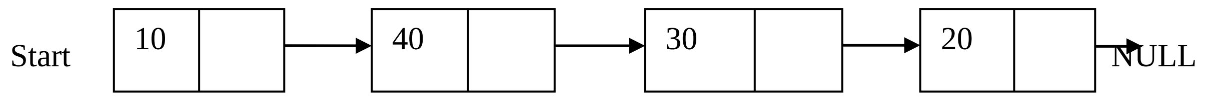


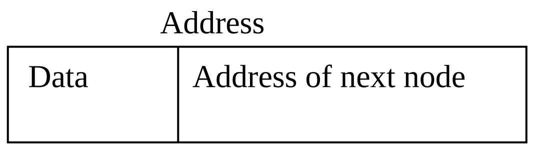
Actual and pictorial representations of linked lists:

Actual representation of linked lists in memory:

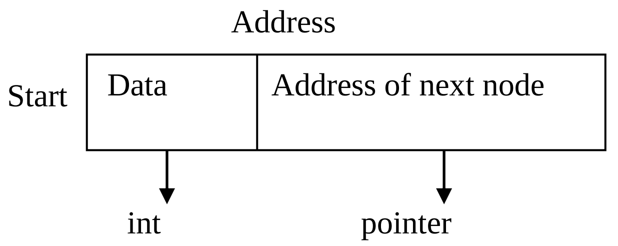


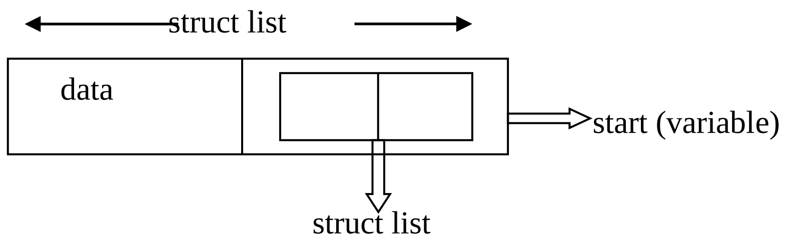
Pictorial representation:





Variable declaration of linked lists

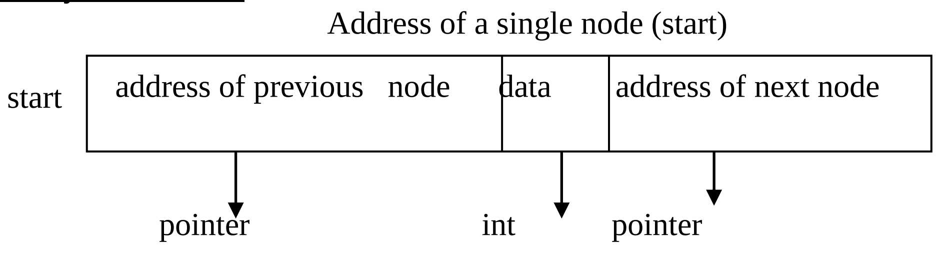


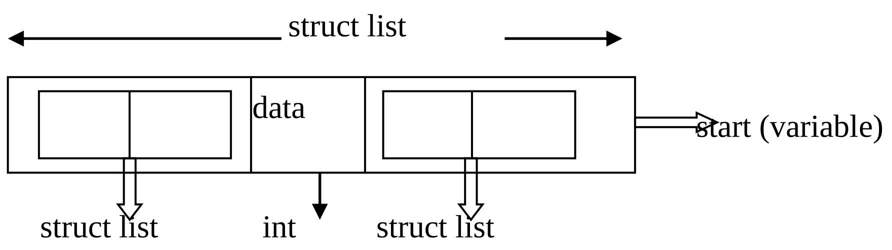


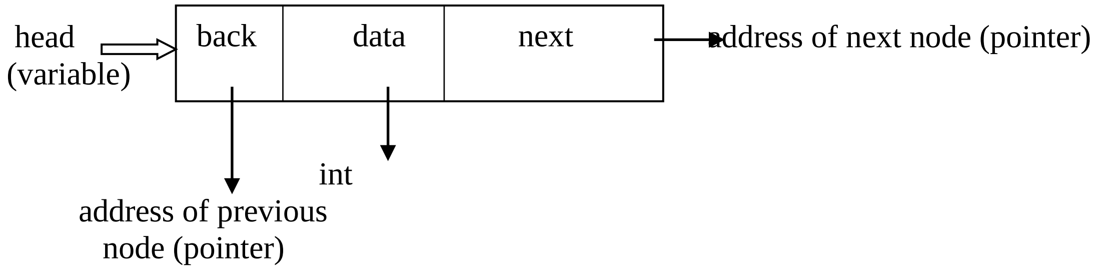
Structure:

```
struct list{
int data;
struct list *next;
};
typedef struct list node;

int main(void) {
node *start;
………………..
………………..
return 0;
}
```

Doubly Linked List:

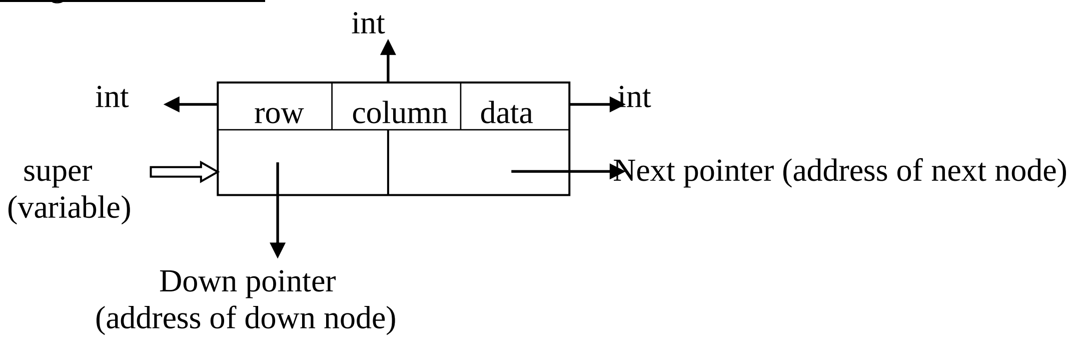






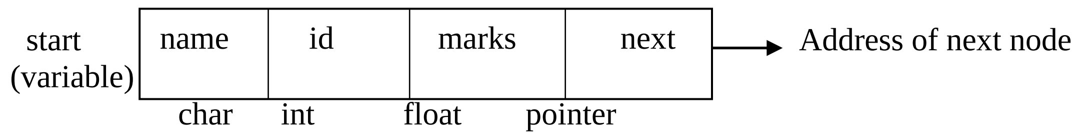
```
struct list {
struct list *back;
int  data;
struct list *next;
};
typedef struct list node;

int main ( ){
node *head;
……………
……………
……………
return 0;
}
```

<u>Orthogonal linked list:</u>



```
struct  list{
int row, col, data;
struct list *down, *right;
};
typedef struct list node;
int main( ) {
node *super;
……………..
……………..
……………..
return 0;
}
```

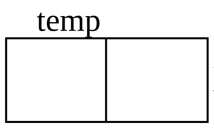
<u>Linear linked list with multiple field:</u>



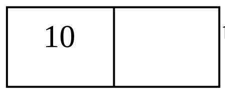
```
struct list{
char name [100];
int id;
float marks;
struct list *next;
};
typedef struct list node;

int main( ){
node *start;
……………
……………
……………
return 0;
}
```

Basic operations using linked lists

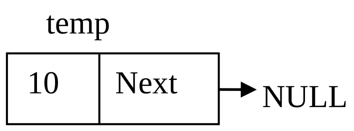
Several statement of linked list:



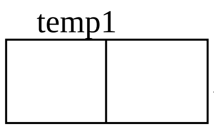
```
int *temp;
temp=(int*)malloc(sizeof(int));
```



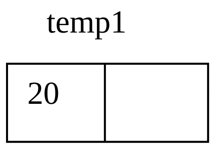
```
temp ->data=10;
```



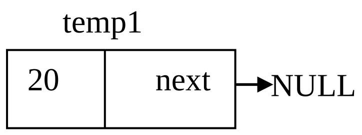
```
temp->next=NULL;
```



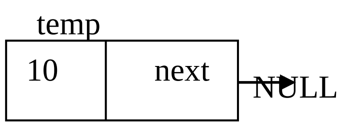
```
int *temp1;
temp1=(int*)malloc (sizeof(int));
```

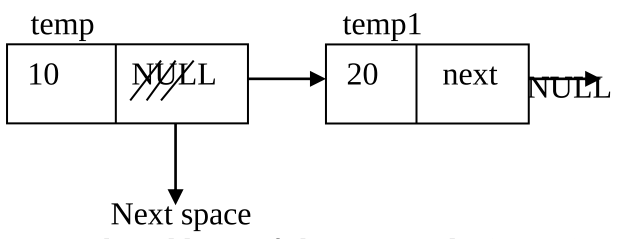


```
temp1->data=20;
```



```
temp1->next=NULL;
```

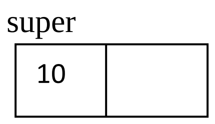




```
temp->next=temp1;
```

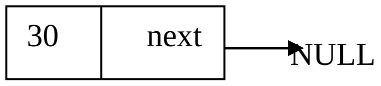
We have to put the address of the next node (temp1) in the ‘next’ space of the previous node (temp).

Reading data from keyboard for a node:



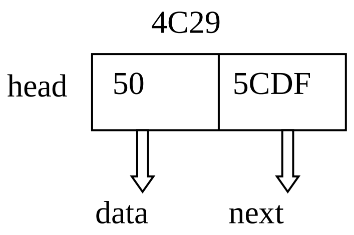
scanf("%d",&super->data);

Writing data from node:



printf("%d",head->data);

Writing data and address from a node:



printf("%d",head->data);

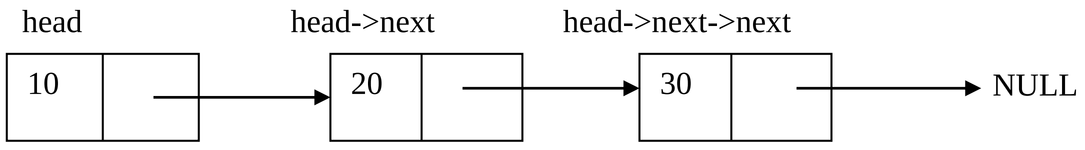
printf("\n%x",& head->next);

printf("\n%x",&head);

Output on monitor:

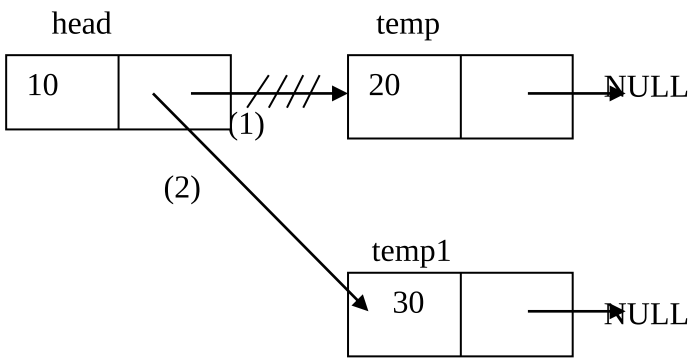
```
50
5CDF
4C29
```

Statements for a linked list:

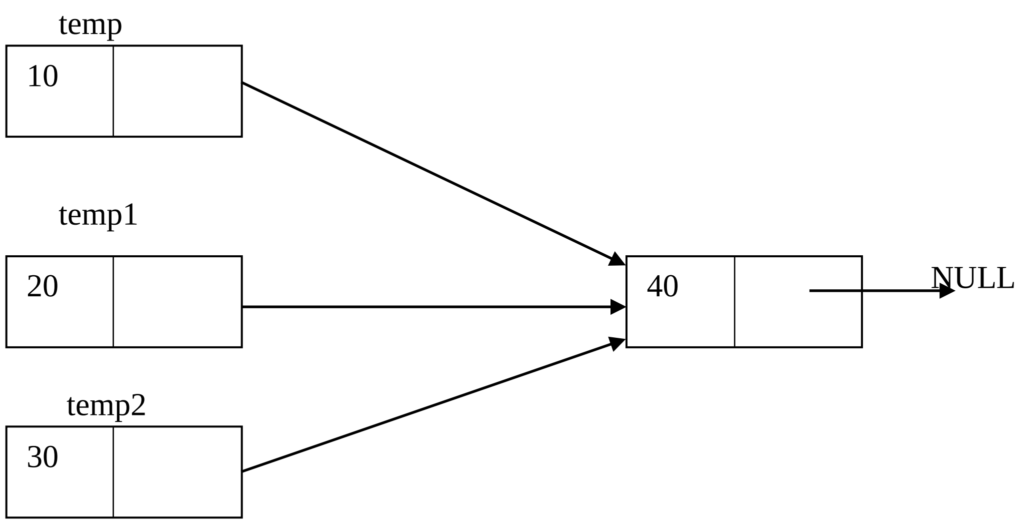


head->data=10; head->next->data=20; head->next->next->data=30;

head->next->next->next=NULL;



One node can point only one node but one node can be pointed by several nodes. Connection (1) is first connected to another node and (2) is second. Connection node ‘temp’ will be disconnected due to connect node ‘temp1’.



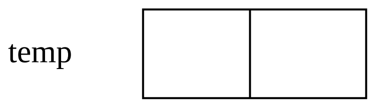
```
printf(“  %d”,temp-> next->data);

printf(“ \n %d”,temp1-> next->data);

printf(“ \n %d”,temp2->next->data);
```

display on monitor:

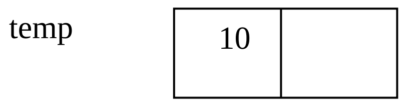
```
40
40
40
```

Statements of connecting two nodes:

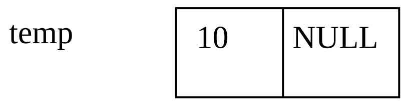
temp=(node*) malloc (sizeof(node));



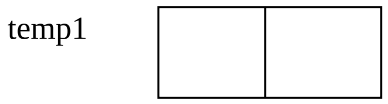
temp->data=10;



temp->NULL;

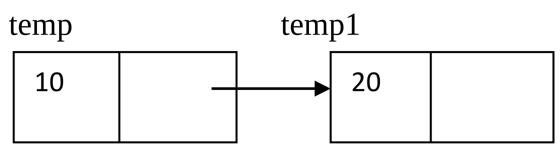


temp1=(node*) malloc(sizeof(node));



temp1->data=20;

temp->next=temp1;



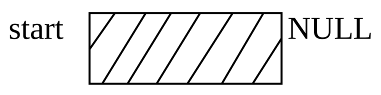
Operation on linear linked lists

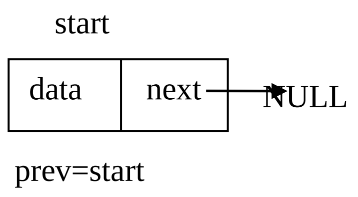
There are several types of operation in linear linked list. They are as follows:

- ✓ Creation
- ✓ Display
- ✓ Insertion
- ✓ Deletion

Creation

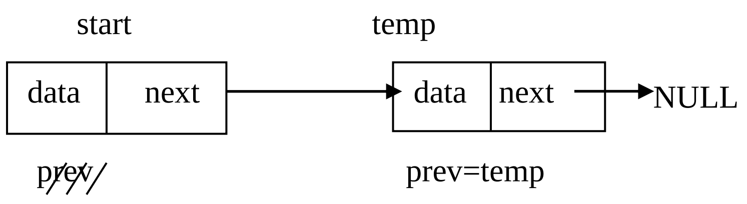
(No node that means start= =NULL)





```
if(start= =NULL)
     {
       start=(node*)malloc(sizeof(node));
       fflush(stdout);
       printf("\n Enter a data:=");
       fflush(stdin);
       scanf("%d",&start->data);
       start->next=NULL;
       prev=start;
     }
```

(At least one node (temp) that means start! =NULL)



```
if(start!=NULL)
     {
        temp=(node*)malloc(sizeof(node));
        fflush(stdout);
        printf("\n Enter a data:=");
        fflush(stdin);
        scanf("%d",&temp->data);
        temp->next=NULL;
        prev->next=temp;
        prev=temp;
      }
```

Write a program in C to create a linear linked list in memory and to display its contents:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
};
typedef struct list node;
int main( )
{
        node *start,*prev,*temp,*temp1;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
                                start->next=NULL;
                                prev=start;
                        }
                        else
                        {
                        temp=(node*)malloc(sizeof(node));
                        fflush(stdout);
                        printf("\n Enter a data:=");
                        fflush(stdin);
                        scanf("%d",&temp->data);
                        temp->next=NULL;
                        prev->next=temp;
                        prev=temp;
                        }
                }
        }
        while(ans=='Y');
        printf("\n Created list:=");
        temp1=start;
        while(temp1!=NULL)
                {
                printf("  %d",temp1->data);
                temp1=temp1->next;
                }
        printf("\n");
        printf("\n");
        return 0;
}
```

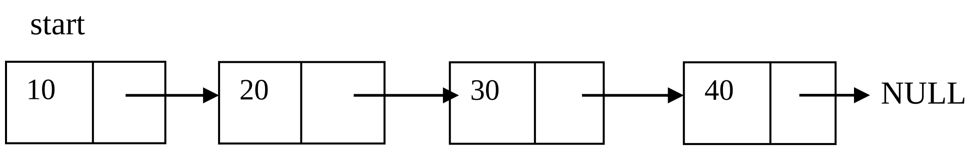
Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
```

Note: fflush (stdout): To avoid garbage value from output buffer

fflush(stdin): To avoid garbage value from input buffer

Display:

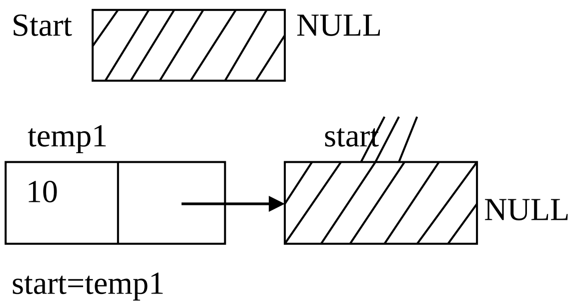


```
int display (node *head) {
            node *temp;
            temp=head;
                while(temp!=NULL)
                        {
                        printf("  %d",temp->data);
                     temp=temp->next;
                        }
return 0;
}

int main( ){
……………….
display(start);
return 0;
}
```

Insertion:

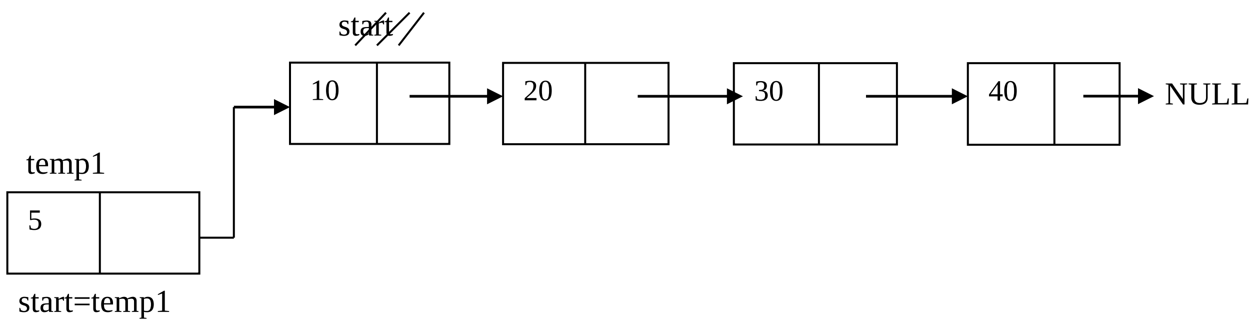
First insertion in a linear linked list:

(No Linked list; start = =NULL, newitem= 10)



```
if(start = =NULL)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=start;
start=temp1;
}
```

<u>First insertion (newitem=5):</u>



```
if(start ->data>=newitem)
{
temp1= (node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=start;
start=temp1;
}
```

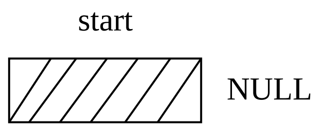
C program for first insertion in a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
};
typedef struct list node;
void display (node *head);
node *insertion(node *start,  int newitem);
int main()
{
        node *start,*prev,*temp;
        int newitem;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
```

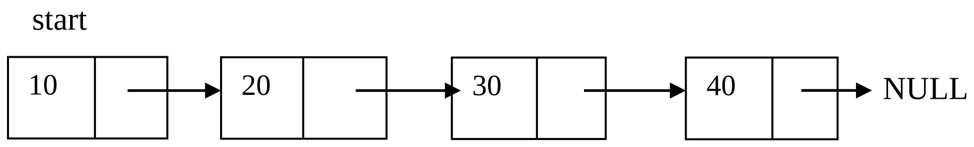
```
                    ans=toupper(getchar());
                    if(ans=='Y')
                    {
                              if(start==NULL)
                              {
                                        start=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&start->data);
                                        start->next=NULL;
                                        prev=start;
                              }
                              else
                              {
                                        temp=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&temp->data);
                                        temp->next=NULL;
                                        prev->next=temp;
                                        prev=temp;
                              }
                    }
          }
          while(ans=='Y');
          display(start);
 do {
          printf("\n Enter a item to be inserted:=");
          scanf("%d",&newitem);
          start=insertion(start,newitem);
          display(start);
          printf("\n");
      } while(1);
          return 0;
}
node *insertion(node *start, int newitem)
{
           node *temp1;
           if((start==NULL)||(start->data>=newitem))
           {
                     temp1=(node*)malloc(sizeof(node));
                     temp1->data=newitem;
                     temp1->next=start;
                     start=temp1;
           }
return start;
}
    void display(node *head)
           {
```

```
        node *temp;
        printf("\n Created list:=");
        temp=head;
           while(temp!=NULL)
                  {
                  printf("  %d",temp->data);
              temp=temp->next;
                  }
        return;
         }
```

Middle insertion (new item=5):



OR



```
if(start= =NULL)
     {
              temp1=(node*)malloc(sizeof(node));
              temp1->data=newitem;
              temp1->next=start;
              start=temp1;
     }
     else if (start->data>newitem)
     {
             printf("\n First value!Enter the middle value!!\n");
             //exit(0);
     }
```

From the figure when there is no linked list we can notice that:

```
start= =NULL
create a node 'temp1'
temp1->data=5;
  temp1->next=NULL;
  start=temp1;
```

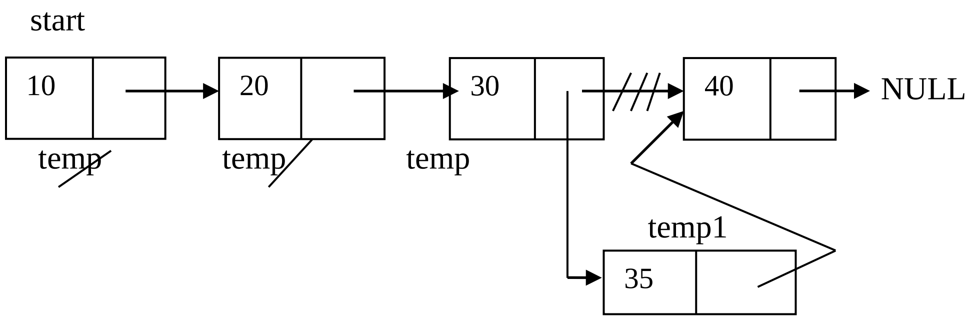
When there is a liked list:

```
start!=NULL
start->data= =10
newitem=5
```

Hence (false || true)= true

That means no first insertion.

<u>Middle insertion (newitem=35):</u>



```
temp=start;
while((temp->next!=NULL)&&(temp->next->data<newitem))
{
temp=temp->next;
}
if((temp->next= =NULL)&&(temp->data<newitem))
{
printf("\n Last value! Enter the middle value!\n");
exit(0);
}
else
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next=temp1;
}
```

C program for middle insertion in a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
};
typedef struct list node;
void display (node *head);
node *insertion(node *start,int newitem);
int main()
{
        node *start,*prev,*temp;
        int newitem;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
                                start->next=NULL;
                                prev=start;
                        }
                        else
                        {
                                temp=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&temp->data);
                                temp->next=NULL;
```

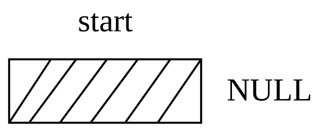
```
                                                prev->next=temp;
                                                prev=temp;
                                    }
                        }
            }
            while(ans=='Y');
            display(start);
do{
            printf("\n Enter a item to be inserted in middle:=");
            scanf("%d",&newitem);
            start=insertion(start,newitem);
            display(start);
             printf("\n");
} while(1);
    return 0;
}
node *insertion(node *start, int newitem){
             node *temp,*temp1;
            if(start==NULL)
             {
                        temp1=(node*)malloc(sizeof(node));
                        temp1->data=newitem;
                        temp1->next=start;
                        start=temp1;
             }
             else if (start->data>newitem)
             {
                       printf("\n First value!Enter the middle value!!\n");
                       //exit(0);
             }
             else
             {
            temp=start;
while((temp->next!=NULL)&&(temp->next->data<newitem))
{
temp=temp->next;
}
if  ((temp->next= =NULL)&&(temp->data<newitem))
           {
            printf("\n Last value! Enter the middle value!\n");
            //exit(0);
            }
else
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next=temp1;
}
             }
return start;
```

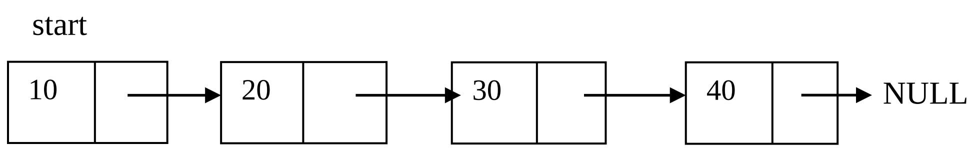
```
}
   void display(node *head)
           {
          node *temp;
          printf("\n Created list:=");
          temp=head;
             while(temp!=NULL)
                   {
                   printf("  %d",temp->data);
                temp=temp->next;
                   }
          return;
           }
```

```
Do you want to create a node(Y/N):=y
Enter a data:=12
Do you want to create a node(Y/N):=y
Enter a data:=13
Do you want to create a node(Y/N):=y
Enter a data:=14
Do you want to create a node(Y/N):=y
Enter a data:=56
Do you want to create a node(Y/N):=n
Created list:=  12  13  14  56
Enter a item to be inserted in middle:=55
Created list:=  12  13  14  55  56
```

<u>Last insertion:</u>

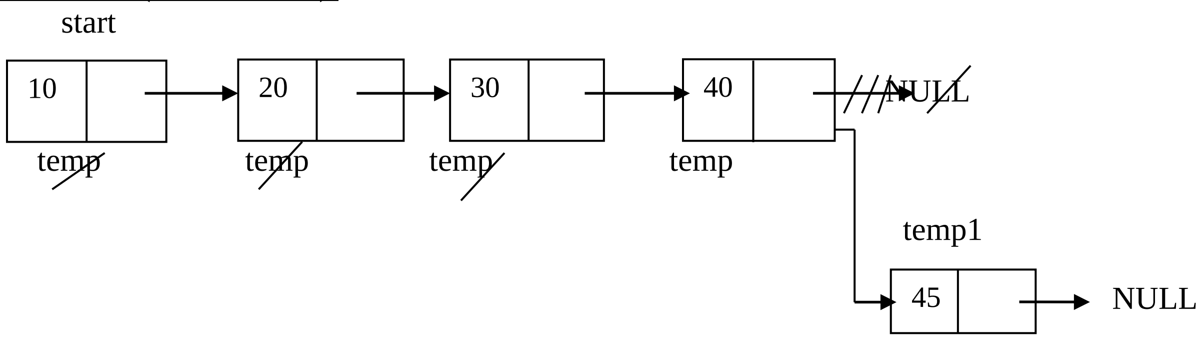
<u>last insertion (start = = null or there is a linked list , newitem=5):</u>





```
 if (start= =NULL)
          {
                   temp1=(node*)malloc(sizeof(node));
                   temp1->data=newitem;
                   temp1->next=start;
                   start=temp1;
          }
          else if (start->data>newitem)
          {
                  printf("\n First value!Enter the Last value!!\n");
                  //exit(0);
          }
```

Last insertion (newitem=45):



```
else  {

 temp=start;
while(temp->next!=NULL)
{
temp=temp->next;
}
if (temp->data<=newitem)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next=temp1;
}
else
{
printf("\n Enter the Last value to be inserted!!\n");
exit(0);
}
}
```

Write a program in C for Last insertion in a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
	int data;
	struct list *next;
};
typedef struct list node;
void display (node *head);
node *insertion(node *start, int newitem);
int main( )
{
	node *start,*prev,*temp;
	int newitem;
	char ans;
	start=NULL;
	do
	{
		fflush(stdout);
		printf("\n Do you want to create a node(Y/N):=");
		fflush(stdin);
		ans=toupper(getchar());
		if(ans=='Y')
		{
			if(start==NULL)
			{
				start=(node*)malloc(sizeof(node));
				fflush(stdout);
				printf("\n Enter a data:=");
				fflush(stdin);
				scanf("%d",&start->data);
				start->next=NULL;
				prev=start;
			}
			else
			{
				temp=(node*)malloc(sizeof(node));
				fflush(stdout);
				printf("\n Enter a data:=");
				fflush(stdin);
				scanf("%d",&temp->data);
				temp->next=NULL;
				prev->next=temp;
				prev=temp;
			}
		}
	}
```

```
            while(ans=='Y');
            display(start);
            do{
            printf("\n Enter a item to be inserted in the last:=");
            scanf("%d",&newitem);
            start=insertion(start,newitem);
            display(start);
            printf("\n");
            } while(1) ;
            return 0;
}
node *insertion(node *start, int newitem)
{
             node *temp,*temp1;
             if(start==NULL)
             {
                        temp1=(node*)malloc(sizeof(node));
                        temp1->data=newitem;
                        temp1->next=start;
                        start=temp1;
             }
             else if (start->data>newitem)
             {
                printf("\n First value! Enter the Last value!!\n");
               //exit(0);
             }
             else
             {
            temp=start;
           while(temp->next!=NULL)
             {
                temp=temp->next;
             }
               if(temp->data<=newitem)
              {
                 temp1=(node*)malloc(sizeof(node));
                 temp1->data=newitem;
                 temp1->next=temp->next;
                 temp->next=temp1;
                }
             else
               {
                 printf("\n Middle value! Enter the Last value !!\n");
                 //exit(0);
               }
               }
return start;
}
     void display(node *head)
             {
            node *temp;
            printf("\n Created list:=");
            temp=head;
               while(temp!=NULL)
                      {
                      printf("  %d",temp->data);
                       temp=temp->next;
                      }
            return;
             }
```

Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=n
Created list:=  10  20  30
Enter a item to be inserted in the last:=35
Created list:=  10  20  30  35
```

Insertion logic (combination of first, middle and last item insertion):

```
  if((start = =NULL)||(start->data>=newitem))

{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=start;
start=temp1;
}
else
{
temp=start;
while ((temp->next!=NULL)&&(temp->next->data<=newitem))
{
temp=temp->next;
}
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next=temp1;
}
```

Write a program in C for First/Middle/Last insertion in a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
};
typedef struct list node;
void display (node *head);
node *insertion(node *start,int newitem);
int main()
{
        node *start,*prev,*temp;
        int newitem;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
```

```
                                        fflush(stdin);
                                        scanf("%d",&start->data);
                                        start->next=NULL;
                                        prev=start;
                                }
                                else
                                {
                                        temp=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&temp->data);
                                        temp->next=NULL;
                                        prev->next=temp;
                                        prev=temp;
                                }
                        }
                }
                while(ans=='Y');
                display(start);
                do{
                printf("\n Enter a item to be inserted(F/M/L:=");
                scanf("%d",&newitem);
                start=insertion(start,newitem);
                display(start);
                printf("\n");
                }while(1);
                return 0;
}
node *insertion(node *start, int newitem)
{
                 node *temp,*temp1;
                 if((start==NULL)||(start->data>=newitem))
                 {
                         temp1=(node*)malloc(sizeof(node));
                         temp1->data=newitem;
                         temp1->next=start;
                         start=temp1;
                 }
                 else
                 {
                temp=start;
while((temp->next!=NULL)&&(temp->next->data<=newitem))
{
temp=temp->next;
}
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next=temp1;
}
```

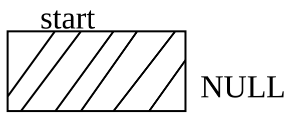
```
return start;
}
    void display(node *head)
             {
            node *temp;
            printf("\n Created list:=");
            temp=head;
               while(temp!=NULL)
                      {
                      printf("  %d",temp->data);
                      temp=temp->next;
                      }
            return;
             }
```

Output:

```
Do you want to create a node(Y/N):=y
Enter a data:=12
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=n
Created list:=  12  20
Enter a item to be inserted(F/M/L:=5
Created list:=  5  12  20
Enter a item to be inserted(F/M/L:=15
Created list:=  5  12  15  20
Enter a item to be inserted(F/M/L:=25
Created list:=  5  12  15  20  25
```

Deletion

No linked list(start==NULL)

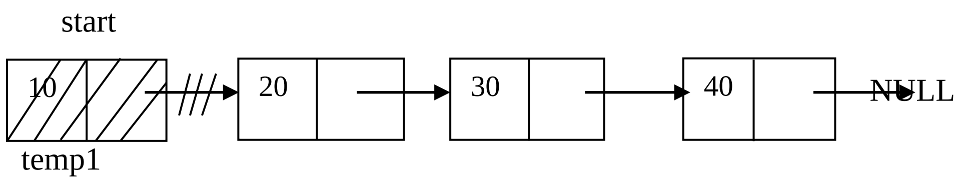


if(start= =NULL){

printf("\nNo linked list");

}

First deletion(item=10)



```
if(start->data= =item) {
temp1=start;
start=start->next;
free(temp1)
}
```

Write a program in C for First deletion from a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
};
typedef struct list node;
void display (node *head);
node *deletion(node *start,int item);
int main( )
{
        node *start,*prev,*temp;
        int item;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
```

```
                                        scanf("%d",&start->data);
                                        start->next=NULL;
                                        prev=start;
                              }
                              else
                              {
                                        temp=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&temp->data);
                                        temp->next=NULL;
                                        prev->next=temp;
                                        prev=temp;
                              }
                    }
          }
          while(ans=='Y');
          display(start);
          do{
          printf("\n Enter first item to be deleted:=");
          scanf("%d",&newitem);
          start=deletion(start,item);
          display(start);
          printf("\n");
          }while(1);
          return 0;
}
node *deletion(node *start, int item)
{
           node *temp1;
           if(start==NULL)
           {
                     printf("\n No linked List");
           }
           else if(start->data==item)
           {
                     temp1=start;
                     start=start->next;
                     free(temp1);
           }
           else
           {
          printf("\nNo Item!Enter the first item to be deleted:");
           }

return start;
}
    void display(node *head)
           {
          node *temp;
```

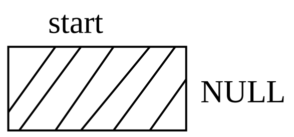
```
            printf("\n Created list:=");
            temp=head;
               while(temp!=NULL)
                      {
                      printf("  %d",temp->data);
                      temp=temp->next;
                      }
            return;
             }
```

Output on monitor:

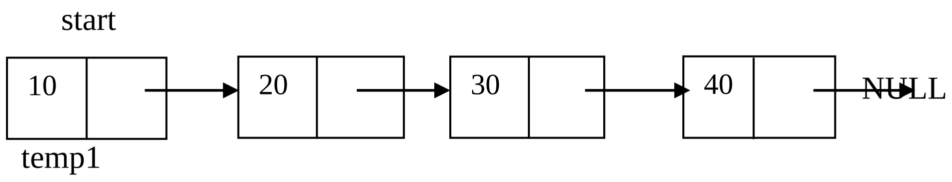
```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter first item to be deleted:=10
Created list:=  20  30  40
```

Middle item deletion:

No linked list(start= =NULL)

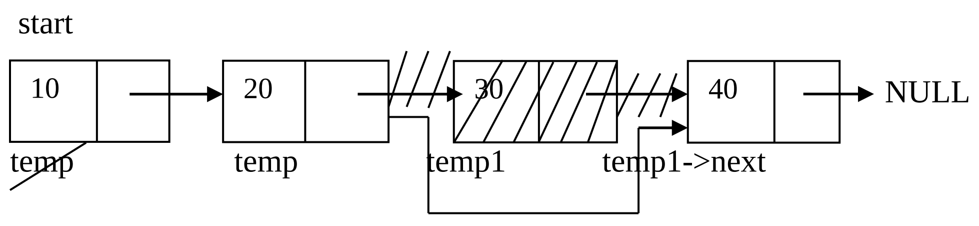


Middle item deletion(item=10)



if((start= =NULL)||(start->data= =item)){
printf("\nNULL or First item!Enter middle item!");   }

Middle item deletion(item=30)



```
temp=start;
while((temp->next!=NULL)&&(temp->next->data)!=item))
{
   temp=temp->next;
}
   if (temp->next)= =NULL)
       {
         printf("\Not Found! Enter middle item!");
       }
   else if (temp->next->next==NULL)
       {
          printf("\Last item! Enter middle item!");
        }
    else
       {
temp1=temp->next;
temp->next=temp1->next;
free(temp1);
        }
```

Write a program in C for Middle item deletion in a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
            int data;
            struct list *next;
};
typedef struct list node;
void display (node *head);
node *deletion(node *start,int item);
int main()
{
            node *start,*prev,*temp;
            int item;
            char ans;
            start=NULL;
            do
            {
                        fflush(stdout);
                        printf("\n Do you want to create a node(Y/N):=");
                        fflush(stdin);
                        ans=toupper(getchar());
                        if(ans=='Y')
                        {
                                    if(start==NULL)
                                    {
                                                start=(node*)malloc(sizeof(node));
                                                fflush(stdout);
                                                printf("\n Enter a data:=");
                                                fflush(stdin);
                                                scanf("%d",&start->data);
                                                start->next=NULL;
                                                prev=start;
                                    }
                                    else
                                    {
                                                temp=(node*)malloc(sizeof(node));
                                                fflush(stdout);
                                                printf("\n Enter a data:=");
                                                fflush(stdin);
                                                scanf("%d",&temp->data);
                                                temp->next=NULL;
                                                prev->next=temp;
                                                prev=temp;
                                    }
                        }
            }
            while(ans=='Y');
            display(start);
            do{
            printf("\n Enter Middle item to be deleted:=");
            scanf("%d",&item);
    start=deletion(start,item);
    display(start);
            printf("\n");
            }while(1);
            return 0;
}
node *deletion(node *start, int item)
{
             node *temp1,*temp;
             if((start==NULL)||(start->data==item))
             {
              printf("\nNULL or First item!!Enter middle item!");
             }
             else
             {
```

```
             temp=start;
while((temp->next!=NULL)&&(temp->next->data!=item))
{
temp=temp->next;
}
if(temp->next==NULL)
{
printf("\n Not Found!!Enter middle item!");
}
else if (temp->next->next==NULL)
{
printf("\n Last item!!Enter middle item!");
}
else
{
temp1=temp->next;
temp->next=temp1->next;
free(temp1);
}
             }
return start;
}
     void display(node *head)
             {
            node *temp;
            printf("\n Created list:=");
            temp=head;
                while(temp!=NULL)
                        {
                        printf("  %d",temp->data);
           temp=temp->next;
                        }
            return;
             }
```

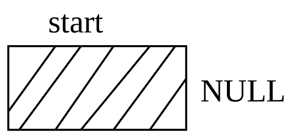
Output on Monitor:

```
 Do you want to create a node(Y/N):=y
 Enter a data:=10
 Do you want to create a node(Y/N):=y
 Enter a data:=20
 Do you want to create a node(Y/N):=y
 Enter a data:=30
 Do you want to create a node(Y/N):=y
 Enter a data:=40
 Do you want to create a node(Y/N):=n
 Created list:=  10  20  30  40
 Enter Middle item to be deleted:=10
NULL or First item!!Enter middle item
 Created list:=  10  20  30  40
```

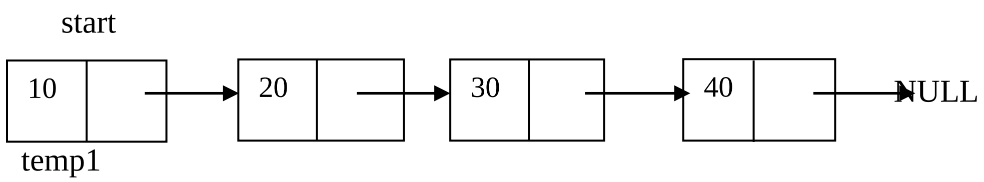
```
Enter Middle item to be deleted:=40
Last item!!Enter middle item!
Created list:=  10  20  30  40
Enter Middle item to be deleted:=50
Not Found!!Enter middle item!
Created list:=  10  20  30  40
Enter Middle item to be deleted:=30
Created list:=  10  20  40
```

Last item deletion:

No linked list(start= =NULL)



Last item deletion(item=10)

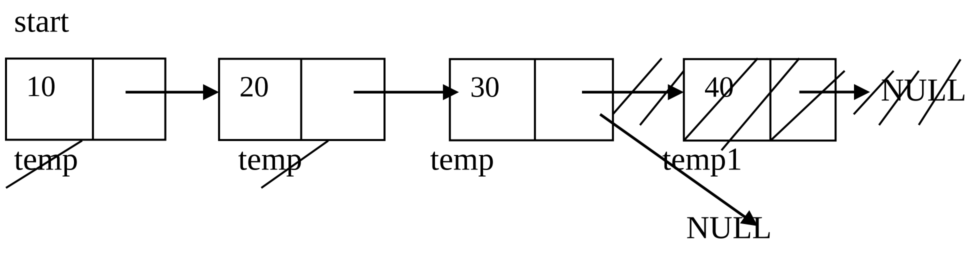


if((start= =NULL)||(start->data= =item)){

printf("\nNULL or First item!Enter middle item!");

}

Last item deletion(item=40)



```
temp=start;
while((temp->next!=NULL)&&(temp->next->data)!=item))
{
   temp=temp->next;
}
   if (temp->next= =NULL)
        {
          printf("\Not Found! Enter Last item!");
        }
    else if(temp->next->next!=NULL)
        {
           printf("\Middle item! Enter Last item!");
        }
    else
        {
temp1=temp->next;
temp->next=temp1->next;
free(temp1);
         }
```

Write a program in C for Last deletion in a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
            int data;
            struct list *next;
};
typedef struct list node;
void display (node *head);
node *deletion(node *start,int item);
int main()
{
            node *start,*prev,*temp;
            int item;
            char ans;
            start=NULL;
            do
            {
                        fflush(stdout);
                        printf("\n Do you want to create a node(Y/N):=");
                        fflush(stdin);
                        ans=toupper(getchar());
                        if(ans=='Y')
                        {
                                    if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
                                start->next=NULL;
                                prev=start;
                        }
                        else
                        {
                                temp=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&temp->data);
                                temp->next=NULL;
                                prev->next=temp;
                                prev=temp;
                        }
                }
        }
        while(ans=='Y');
        display(start);
        do{
        printf("\n Enter Last item to be deleted:=");
        scanf("%d",&item);
        start=deletion(start,item);
        display(start);
        printf("\n");
        }while(1);
        return 0;
}
node *deletion(node *start, int item)
{
         node *temp1,*temp;
         if((start==NULL)||(start->data==item))
         {
          printf("\nNULL or First item!!Enter Last item!");
         }
         else
         {
         temp=start;
while((temp->next!=NULL)&&(temp->next->data!=item))
{
temp=temp->next;
}
if(temp->next==NULL)
{
printf("\n Not Found!!Enter Last item!");
}
```

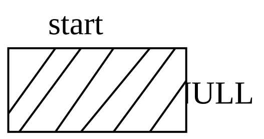
```
else if(temp->next->next!=NULL)
{
            printf("\n Middle item!!Enter Last item!");
}
else
{
temp1=temp->next;
temp->next=temp1->next;
free(temp1);
}
             }
return start;
}
     void display(node *head)
             {
            node *temp;
            printf("\n Created list:=");
            temp=head;
            while(temp!=NULL)
                        {
                        printf("  %d",temp->data);
                         temp=temp->next;
                        }
            return;
             }
```

Output on monitor:

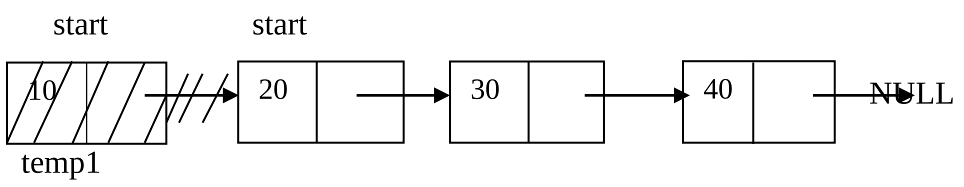
```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter Last item to be deleted:=10
ULL or First item!!Enter Last item!
Created list:=  10  20  30  40
Enter Last item to be deleted:=50
Not Found!!Enter Last item!
Created list:=  10  20  30  40
Enter Last item to be deleted:=20
Middle item!!Enter Last item!
Created list:=  10  20  30  40
Enter Last item to be deleted:=40
Created list:=  10  20  30
```

Deletion logic (combination of first, middle and last item deletion):

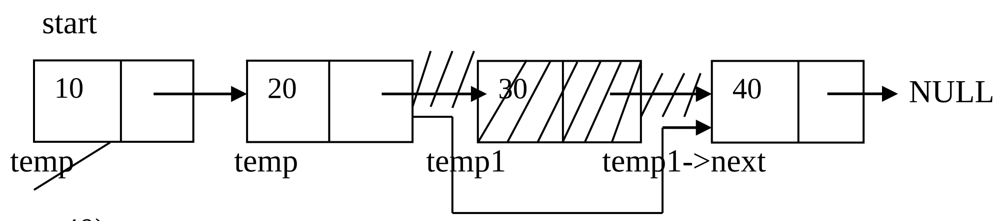
No linked list(start= =NULL)



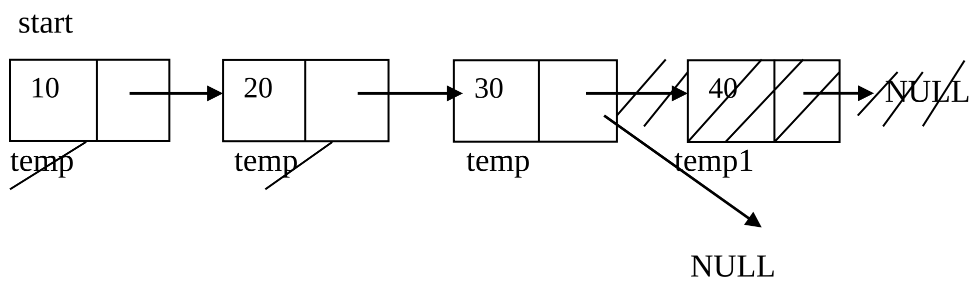
First item deletion(item=10)



Middle item deletion(item=30)



Last item deletion(item=40)



```
if(start= =NULL)
{
 printf("\n No linked list!!");
}
else if(start->data= =item)
{
temp1=start;
start=start->next;
free(temp1)
}
else
{
```

```
temp=start;
while((temp->next!=NULL)&&(temp->next->data)!=item))
    {
      temp=temp->next;
    }
     if (temp->next= =NULL)
         {
           printf("\Not Found! Enter the item!");
         }
    else
         {
temp1=temp->next;
temp->next=temp1->next;
free(temp1);
          }
}
```

Write a program in C for First/Middle/Last deletion in a linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
            int data;
            struct list *next;
};
typedef struct list node;
void display (node *head);
node *deletion(node *start,int item);
int main()
{
            node *start,*prev,*temp;
            int item;
            char ans;
            start=NULL;
            do
            {
                        fflush(stdout);
                        printf("\n Do you want to create a node(Y/N):=");
                        fflush(stdin);
                        ans=toupper(getchar());
                        if(ans=='Y')
                        {
                                    if(start==NULL)
                                    {
                                                start=(node*)malloc(sizeof(node));
                                                fflush(stdout);
```

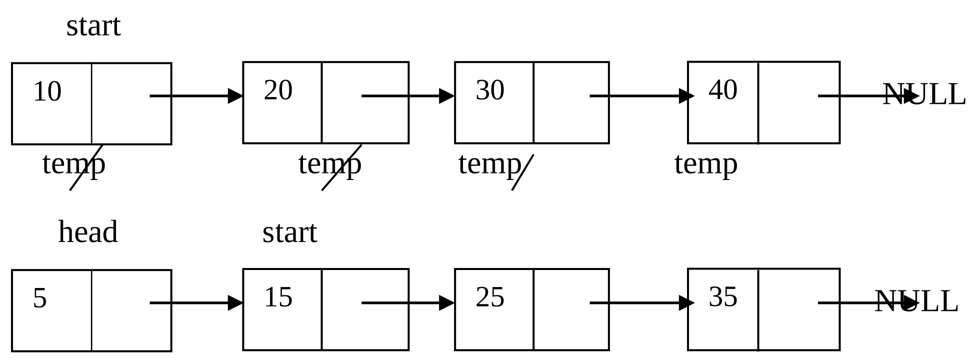
```
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&start->data);
                                        start->next=NULL;
                                        prev=start;
                              }
                              else
                              {
                                        temp=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&temp->data);
                                        temp->next=NULL;
                                        prev->next=temp;
                                        prev=temp;
                              }
                    }
          }
          while(ans=='Y');
          display(start);
          do{
          printf("\n Enter the item to be deleted:=");
          scanf("%d",&item);
   start=deletion(start,item);
   display(start);
          printf("\n");
          }while(1);
          return 0;
}
node *deletion(node *start, int item)
{
          node *temp1,*temp;

          if(start==NULL)
          {
           printf("\nNo linked list!");
          }
          else if (start->data==item)
          {
                    temp1=start;
                    start=start->next;
                    free(temp1);
          }
    else
          {

          temp=start;
while((temp->next!=NULL)&&(temp->next->data!=item))
{
temp=temp->next;
```

```
}
if(temp->next==NULL)
{
printf("\n Not Found!!Enter the item!");

}
else
{
temp1=temp->next;
temp->next=temp1->next;
free(temp1);
}
             }
return start;
}
     void display(node *head)
             {
            node *temp;
            printf("\n Created list:=");
            temp=head;
                while(temp!=NULL)
                        {
                        printf("  %d",temp->data);
           temp=temp->next;
                        }
            return;
             }
```

Output on monitor:

```
Do you want to create a node(Y/N):
Enter a data:=10
Do you want to create a node(Y/N):
Enter a data:=20
Do you want to create a node(Y/N):
Enter a data:=30
Do you want to create a node(Y/N):
Enter a data:=40
Do you want to create a node(Y/N):n
Created list:=  10  20  30  40
Enter a item(F/M/L)to be deleted:=10
Created list:=  20  30  40
Enter a item(F/M/L)to be deleted:=30
Created list:=  20  40
Enter a item(F/M/L)to be deleted:=42
Not Found!!Enter the item!
Created list:=  20  40
Enter a item(F/M/L)to be deleted:=40
Created list:=  20
```

Concatenation



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

Write a C program to concatenate two linked list and display the data after concatenation.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
};
typedef struct list node;
void display (node *head);
int main()
{
        node *start,*prev,*temp,*head;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N) in first list:=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
```

```
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&start->data);
                                        start->next=NULL;
                                        prev=start;
                                }
                                else
                                {
                                        temp=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&temp->data);
                                        temp->next=NULL;
                                        prev->next=temp;
                                        prev=temp;
                                }
                        }
                }
                while(ans=='Y');
                display(start);
                printf("\n");
                head=NULL;
                do      {
                        fflush(stdout);
                        printf("\n Do you want to create a node(Y/N)in second list:=");
                        fflush(stdin);
                        ans=toupper(getchar());
                        if(ans=='Y')
                        {
                                if(head==NULL)
                                {
                                        head=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&head->data);
                                        head->next=NULL;
                                        prev=head;
                                }
                                else
                                {
                                        temp=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&temp->data);
                                        temp->next=NULL;
                                        prev->next=temp;
                                        prev=temp;
                                }
```

```
                    }
            }
            while(ans=='Y');
            display(head);
            printf("\n");
            temp=start;
            while(temp->next!=NULL)
            {
                    temp=temp->next;
            }
            temp->next=head;
            display(start);
            printf("\n");
            return 0;
}
     void display(node *head)       {
            node *temp;
            printf("\n Created list:=");
            temp=head;
               while(temp!=NULL)
                    {
                    printf("  %d",temp->data);
            temp=temp->next;
                    }
            return;
             }
```

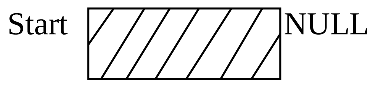
Output on monitor:

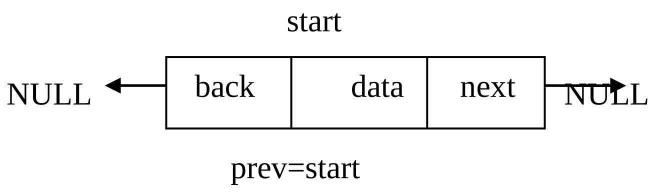
```
Do you want to create a node(Y/N) in first list:=y
Enter a data:=10
Do you want to create a node(Y/N) in first list:=y
Enter a data:=20
Do you want to create a node(Y/N) in first list:=y
Enter a data:=30
Do you want to create a node(Y/N) in first list:=n
Created list:=  10  20  30
Do you want to create a node(Y/N)in second list:=y
Enter a data:=5
Do you want to create a node(Y/N)in second list:=y
Enter a data:=15
Do you want to create a node(Y/N)in second list:=n
Created list:=  5  15
Created list:=  10  20  30  5  15
```

Operation on Doubly Linked Lists

Creation

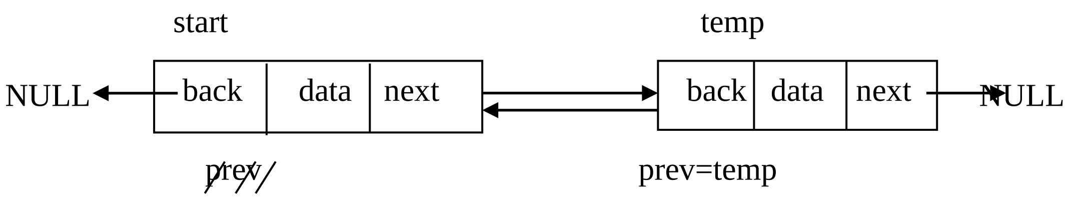
(No node that means start= =NULL)





```
if(start= =NULL)
         {
           start=(node*)malloc(sizeof(node));
           printf("\n Enter a data:=");
           scanf("%d",&start->data);
           start->next=NULL;
         start->back=NULL;
           prev=start;
        }
```

(At least one node (temp) that means start! =NULL)



```
if(start!=NULL)   {
           temp=(node*)malloc(sizeof(node));
           printf("\n Enter a data:=");
           scanf("%d",&temp->data);
           temp->next=NULL;
          temp ->back= prev
           prev->next=temp;
           prev=temp;
        }
```

Reverse contents display logic:

```
temp=start;
        printf("\n Created reverse list:=");
        while(temp->next!=NULL)
        {
                temp=temp->next;
        }
while(temp!=NULL)
{
        printf("  %d",temp->data);
        temp=temp->back;
}
```

Write a program in C to create a doubly linked list in memory and to display its contents in forward and reverse cases:

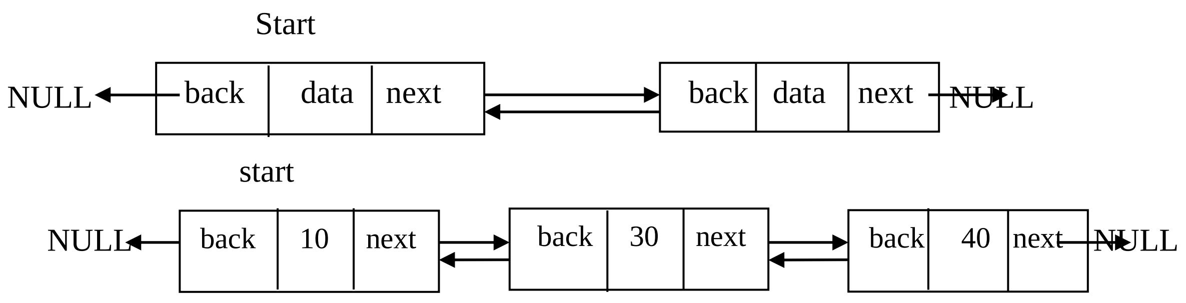
```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
        struct list *back;
};
typedef struct list node;
void display (node *head);
void reverse (node *start);
int main( ){
        node *start,*prev,*temp;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
```

```
                                        start->next=NULL;
                                        start->back=NULL;
                                        prev=start;
                              }
                              else
                              {
                                        temp=(node*)malloc(sizeof(node));
                                        fflush(stdout);
                                        printf("\n Enter a data:=");
                                        fflush(stdin);
                                        scanf("%d",&temp->data);
                                        temp->next=NULL;
                                        prev->next=temp;
                                        temp->back=prev;
                                        prev=temp;
                              }
                    }
          }
          while(ans=='Y');
          display(start);
          printf("\n");
          reverse(start);
          printf("\n");
          return 0;
}
    void display(node *head)      {
          node *temp;
          printf("\n Created list:=");
          temp=head;
             while(temp!=NULL)
                    {
                    printf("  %d",temp->data);
                     temp=temp->next;
                    }
          return;
           }
void reverse (node *start){
          node *temp;
          temp=start;
          printf("\n Created reverse list:=");
          while(temp->next!=NULL)
          {
                    temp=temp->next;
          }
while(temp!=NULL)
{
          printf("  %d",temp->data);
          temp=temp->back;
}
return;
}
```

Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=n
Created list:=  10  20  30
Created reverse list:=  30  20  10
```

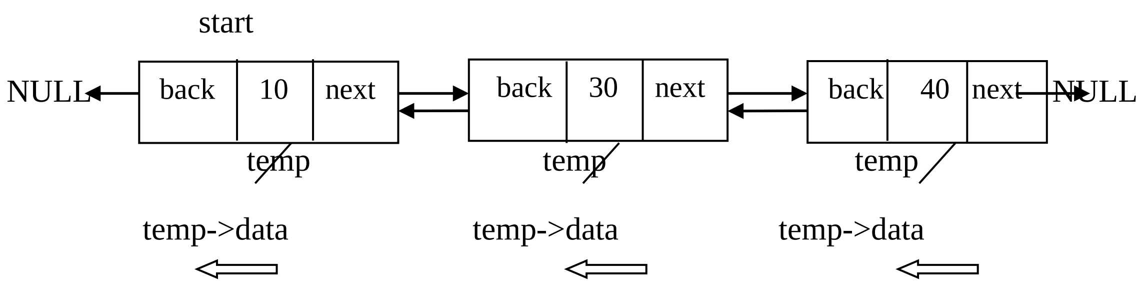
Display



Forward display:

```
display (node *head) {
            node *temp;
            temp=head;
                while(temp!=NULL)
                        {
                        printf("  %d",temp->data);
                     temp=temp->next;
                        }
return;
}
int main( )
{
……………….
display(start);
return 0;
}
```

Reverse display:



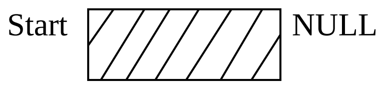
```
void reverse (node *start){
node *temp;
temp=start;
            printf("\n Created reverse list:=");
            while(temp->next!=NULL)
            {
                        temp=temp->next;
            }
while(temp!=NULL)
           {
            printf("  %d",temp->data);
            temp=temp->back;
           }
}
_int main( ) {
……………….
reverse (start);
return 0;
}
```

Insertion:

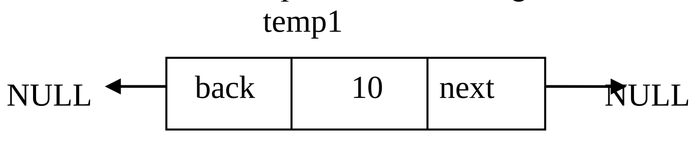
Insertion logic (combination of **first, middle and last item insertion into doubly linked list):**

First insertion (newitem = =10)

(No node that means start= =NULL)



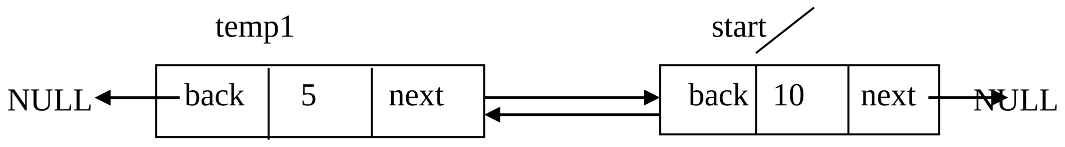
(Create a node name ‘temp1’ and later changes it’s name ‘start’)



start=temp1

```
if(start==NULL)
{
temp1=(node*)malloc(sizeof(node));
 temp1->data=newitem;
 temp1->next=start;
 temp1->back=start;
 start=temp1;
}
```

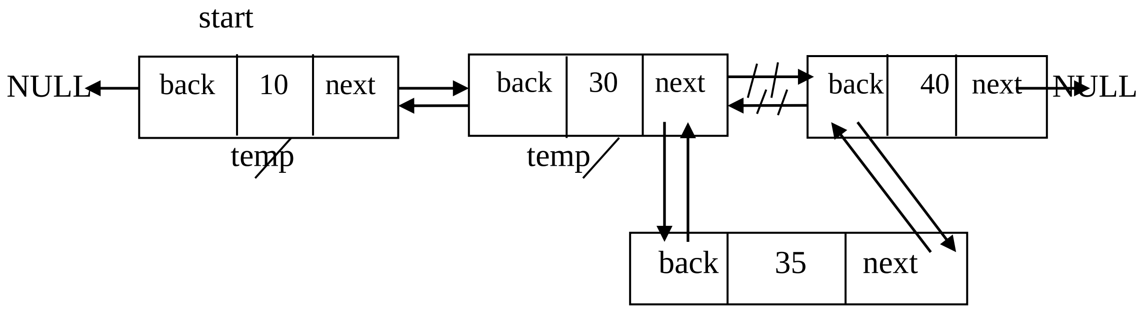
(if start is not NULL, that means there is a node ; newitem= =5)



start=temp1

```
 if (start->data>=newitem)      {
                     temp1=(node*)malloc(sizeof(node));
                     temp1->data=newitem;
                     temp1->next=start;
                     start->back=temp1;
                     temp1->back=NULL;
                     start=temp1;    }
```

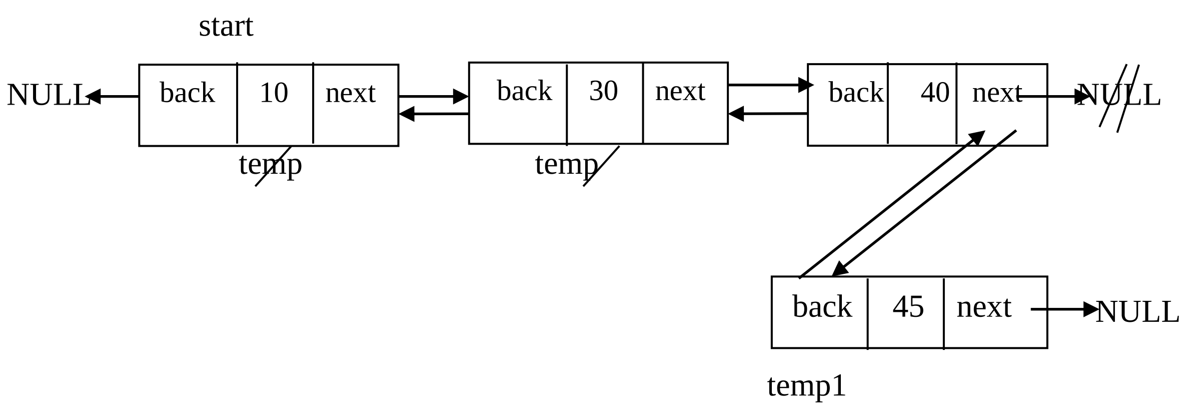
(Middle insertion newitem =35)



temp1

```
temp=start;
while((temp->next!=NULL)&&(temp->next->data<=newitem))
{
temp=temp->next;
}
if(temp->next!=NULL)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next->back=temp1;
temp1->back=temp;
temp->next=temp1;
}
```

(Last insertion newitem =45)



```
temp=start;
while((temp->next!=NULL)&&(temp->next->data<=newitem))
{
temp=temp->next;
}
if(temp->next==NULL)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp1->back=temp;
temp->next=temp1;
}
```

Insertion logic (combination of **first, middle and last item insertion into doubly linked list):**

```
if(start==NULL)
            {
                        temp1=(node*)malloc(sizeof(node));
                        temp1->data=newitem;
                        temp1->next=start;
                        temp1->back=start;
                        start=temp1;
            }
           else if (start->data>=newitem)
            {
                        temp1=(node*)malloc(sizeof(node));
                        temp1->data=newitem;
                        temp1->next=start;
                        start->back=temp1;
                        temp1->back=NULL;
                        start=temp1;
            }
            else
            {
           temp=start;
while((temp->next!=NULL)&&(temp->next->data<=newitem))
{
temp=temp->next;
}
if(temp->next!=NULL)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next->back=temp1;
temp1->back=temp;
temp->next=temp1;
}
else
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp1->back=temp;
temp->next=temp1;
}
            }
```

Write a program in C to insert item first, middle and last into doubly linked list and to display its contents in forward and reverse cases:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
        struct list *back;
};
typedef struct list node;
void display (node *head);
node *insertion(node *start,int newitem);
void reverse (node *start);
int main()
{
        node *start,*prev,*temp;
        int newitem;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
                                start->next=NULL;
          start->back=NULL;
                                prev=start;
                        }
                        else
                        {
        temp=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&temp->data);
                                temp->next=NULL;
                                prev->next=temp;
```

```
                                        temp->back=prev;
                                        prev=temp;
                              }
                    }
          }
          while(ans=='Y');
          display(start);
          //reverse(start);
          do{
          printf("\n Enter a item to be inserted(F/M/L:=");
          scanf("%d",&newitem);
   start=insertion(start,newitem);
   display(start);
          reverse(start);
          printf("\n");
          }while(1);
          return 0;
}
node *insertion(node *start, int newitem)
{
           node *temp,*temp1;
           if(start==NULL)
           {
                     temp1=(node*)malloc(sizeof(node));
                     temp1->data=newitem;
                     temp1->next=start;
                     temp1->back=start;
                     start=temp1;
           }
          else if (start->data>=newitem)
           {
                     temp1=(node*)malloc(sizeof(node));
                     temp1->data=newitem;
                     temp1->next=start;
                     start->back=temp1;
                     temp1->back=NULL;
                     start=temp1;
           }
           else
           {
          temp=start;
while((temp->next!=NULL)&&(temp->next->data<=newitem))
{
temp=temp->next;
}
if(temp->next!=NULL)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp->next->back=temp1;
```

```
temp1->back=temp;
temp->next=temp1;
}
else
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp->next;
temp1->back=temp;
temp->next=temp1;
}
              }
return start;
}
     void display(node *head)
              {
             node *temp;
             printf("\n Created list:=");
             temp=head;
                 while(temp!=NULL)
                          {
                          printf("  %d",temp->data);
            temp=temp->next;
                          }
             return;
              }

 void reverse (node *start)
{
             node *temp;
             temp=start;
             printf("\n Created reverse list:=");
             while(temp->next!=NULL)
             {
                          temp=temp->next;
             }

    while(temp!=NULL)
             {
             printf("  %d",temp->data);
             temp=temp->back;
             }
    return;
}
```

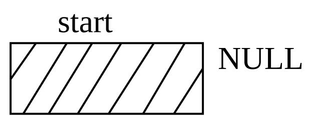
Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter a item to be inserted(F/M/L:=5
Created list:=  5  10  20  30  40
Created reverse list:=  40  30  20  10  5
Enter a item to be inserted(F/M/L:=35
Created list:=  5  10  20  30  35  40
Created reverse list:=  40  35  30  20  10  5
Enter a item to be inserted(F/M/L:=45
Created list:=  5  10  20  30  35  40  45
Created reverse list:=  45  40  35  30  20  10  5
```

<u>Deletion:</u>

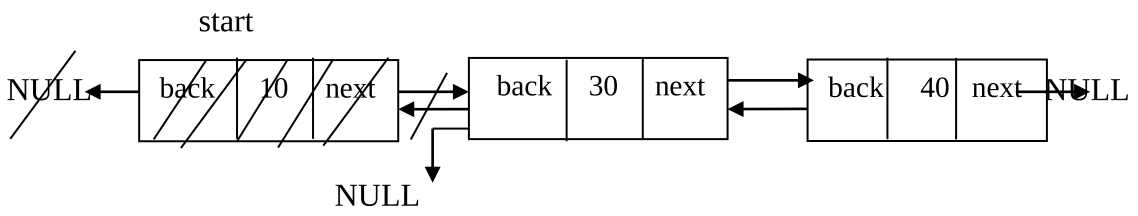
<u>First deletion (item = =10)</u>

<u> (No node that means start= =NULL)</u>



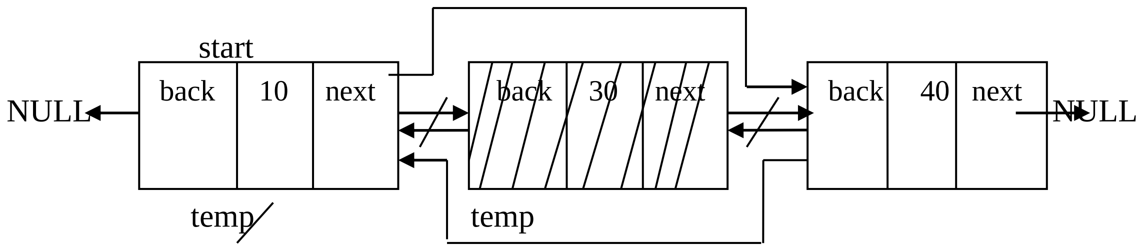
```
if(start==NULL)
            {
             printf("\nNo linked list!");
            }
```

<u>If there is a linked list:</u>



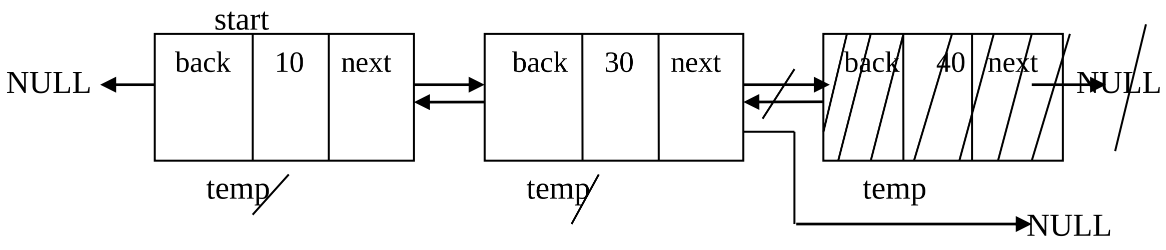
```
if (start->data= =item)
            {
                        start=start->next;
                        free(start->back);
                        start->back=NULL;
            }
```

Middle deletion (item=30)



```
temp=start;
while((temp->next!=NULL)&&(temp->data!=item))
{
temp=temp->next;
}
 if(temp->next!=NULL)
{
temp->back->next=temp->next;
temp->next->back=temp->back;
free(temp);
}
```

Last deletion (item==40)



```
if((temp->next==NULL)&&(temp->data= =item))
{
temp->back->next=temp->next;
free(temp);
}
```

Combination logic of first,middle and last deletion from doubly linked list:

```
if(start= =NULL)
            {
             printf("\nNo linked list!");
            }
            else if (start->data==item)
            {
                        start=start->next;
                        free(start->back);
                        start->back=NULL;
            }
     else
            {
            temp=start;
while((temp->next!=NULL)&&(temp->data!=item))
{
temp=temp->next;
}
 if(temp->next!=NULL)
{
temp->back->next=temp->next;
temp->next->back=temp->back;
free(temp);
}
else if((temp->next= =NULL)&&(temp->data= =item))
{
temp->back->next=temp->next;
free(temp);
}
else
{
printf("\n Not Found!!Enter the item!");
}
            }
```

Write a program in C to delete item of first, middle and last from doubly linked list and to display the contents in forward and reverse cases.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
            int data;
            struct list *next,*back;
};
```

```
typedef struct list node;
void display (node *head);
void reverse(node *start);
node *deletion(node *start,int item);
int main()
{
        node *start,*prev,*temp;
        int item;
        char ans;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
                                start->next=NULL;
                                start->back=NULL;
                                prev=start;
                        }
                        else
                        {
                                temp=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&temp->data);
                                temp->next=NULL;
                                prev->next=temp;
                                temp->back=prev;
                                prev=temp;
                        }
                }
        }
        while(ans=='Y');
        display(start);
        //reverse(start);
        do{
        printf("\n Enter a item(F/M/L)to be deleted:=");
        scanf("%d",&item);
   start=deletion(start,item);
   display(start);
```

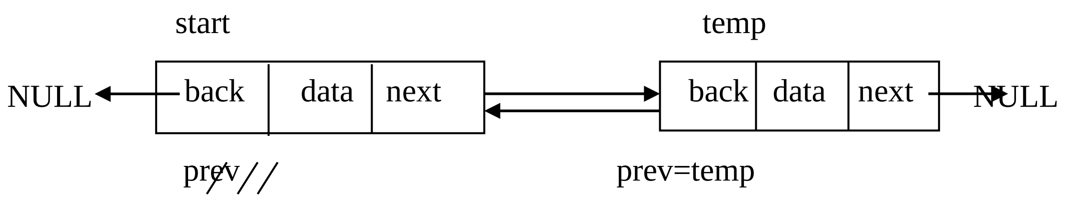
```
            reverse(start);
            printf("\n");
            }while(1);
            return 0;
}
node *deletion(node *start, int item)
{
             node *temp;
             if(start==NULL)
             {
              printf("\nNo linked list!");
             }
             else if (start->data==item)
             {
                         start=start->next;
                         free(start->back);
                         start->back=NULL;
             }
    else
             {
             temp=start;
while((temp->next!=NULL)&&(temp->data!=item))
{
temp=temp->next;
}
 if(temp->next!=NULL)
{
temp->back->next=temp->next;
temp->next->back=temp->back;
free(temp);
}
else if((temp->next==NULL)&&(temp->data==item))
{
temp->back->next=temp->next;
free(temp);
}
else
{
printf("\n Not Found!!Enter the item!");
}
             }
return start;
}
    void display(node *head)
             {
            node *temp;
            printf("\n Created list:=");
            temp=head;
                while(temp!=NULL)
                        {
                        printf("  %d",temp->data);
```

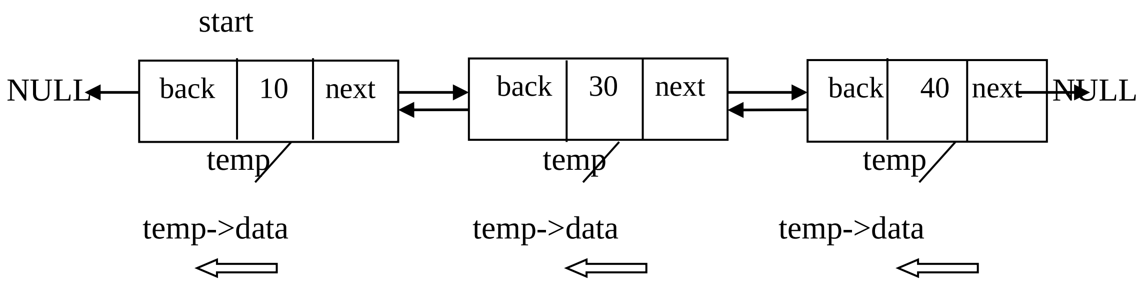
```
        temp=temp->next;
                }
         return;
         }
         void reverse(node *start)
         {
                 node *temp;
                 printf("\n Created reverse list:=");

    temp=start;
                 while(temp->next!=NULL)
                 {
                         temp=temp->next;
                 }
            while(temp!=NULL)
                 {
             printf("  %d",temp->data);
             temp=temp->back;
                 }
 return;
         }
```

Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter a item(F/M/L)to be deleted:=10
Created list:=  20  30  40
Created reverse list:=  40  30  20
Enter a item(F/M/L)to be deleted:=30
Created list:=  20  40
Created reverse list:=  40  20
Enter a item(F/M/L)to be deleted:=40
Created list:=  20
Created reverse list:=  20
Enter a item(F/M/L)to be deleted:=50
Not Found!!Enter the item!
Created list:=  20
Created reverse list:=  20
```

Reverse



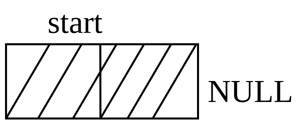


```
temp=start;
          printf("\n Created reverse list:=");
          while(temp->next!=NULL)    {
                    temp=temp->next;
          }
while(temp!=NULL)          {
          printf("  %d",temp->data);
          temp=temp->back;
         }
```

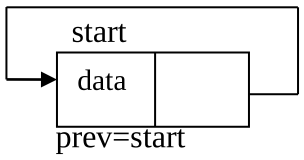
Circular Linked Lists

Creation of circular linked lists:

If start= =NULL that means no linked list:

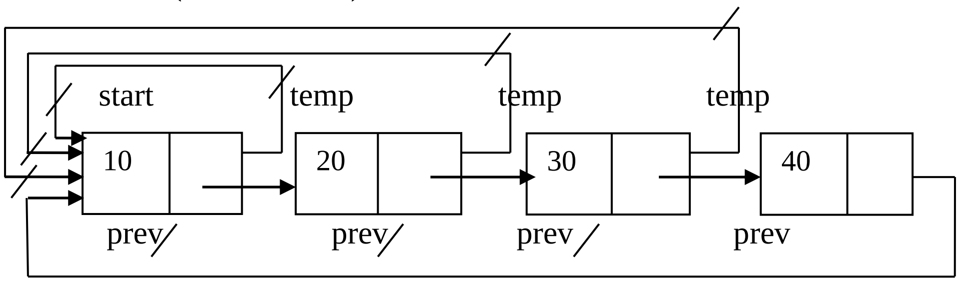


(Need to create a node named ‘start’and later assigned it to prev)



```
if(start= =NULL)
      {
                start=(node*)malloc(sizeof(node));
                scanf("%d",&start->data);
                start->next=start;
                prev=start;
      }
```

If there is a node, that means (start!=NULL)

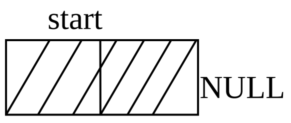


```
  if(start!=NULL)
{
            temp=(node*)malloc(sizeof(node));
            scanf("%d",&temp->data);
            temp->next=start;
            prev->next=temp;
            prev=temp;
  }
```

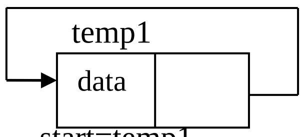
First and Last Insertion

First Insertion:

If start= =NULL that means no linked list:



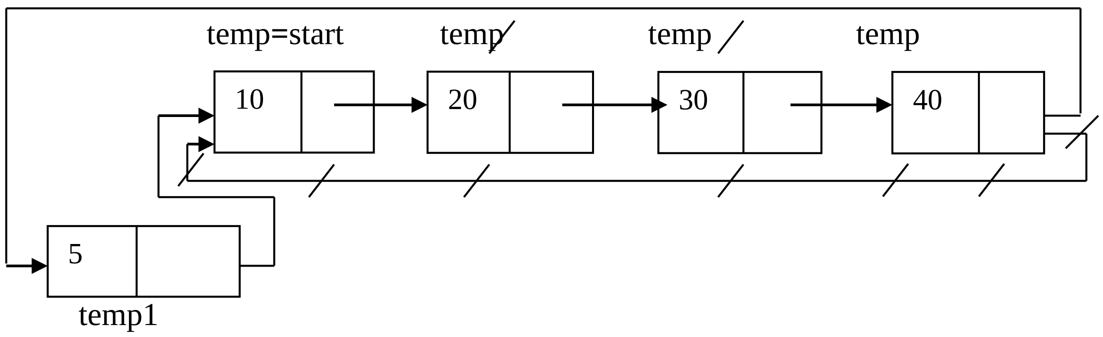
(Need to create a node named ‘temp1’and later assigned it to start)



start=temp1

```
if(start= =NULL)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp1;
start=temp1;
}
```

If there is a linked list; First insertion (newitem=5)



```
temp=start;
while(temp->next!=start)
{
temp=temp->next;
}
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=start;
temp->next=temp1;
start=temp1;
```

Display of circular linked list:

```
temp1=start;
if (temp1!=NULL)       // when no linked lists
{
  do
   {
    printf("  %d",temp1->data);
     temp1=temp1->next;
    } while (temp1! =start);
 }
```

Write a program in C to insert first new item into circular linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
            int data;
            struct list *next;
```

```
};
typedef struct list node;
node *insertion(node *start,int newitem);
void display(node *start);
int main( ) {
        node *start,*prev,*temp;
        char ans;
        int newitem;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
                                start->next=start;
                                prev=start;
                        }
                        else
                        {
                                temp=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&temp->data);
                                temp->next=start;
                                prev->next=temp;
                                prev=temp;
                        }
                }
        }
        while(ans=='Y');
        printf("\n Created list:=");
        display(start);
        printf("\n");
        do{
        printf("\n Enter a item to be inserted(F/M/L:=");
        scanf("%d",&newitem);
        start=insertion(start,newitem);
        printf("\n Created list after first insertion:=");
        display(start);
        printf("\n");
```

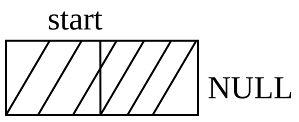
```
            }while(1);
            return 0;
}
node *insertion(node *start,int newitem)
{
   node *temp1,*temp;
            if(start==NULL)
            {
            temp1=(node*)malloc(sizeof(node));
            temp1->data=newitem;
            temp1->next=temp1;
            start=temp1;
            }
            else
            {
            temp=start;
            while(temp->next!=start)
            {
              temp=temp->next;
            }
            temp1=(node*)malloc(sizeof(node));
            temp1->data=newitem;
            temp1->next=start;
            temp->next=temp1;
            start=temp1;
            }
return start;
}
void display(node *start)
{
            node *temp1;
            temp1=start;
            if(temp1!=NULL)       // when no linked lists
            {
            do
                        {
                        printf("  %d",temp1->data);
                         temp1=temp1->next;
                        }while(temp1!=start);
            }
                        return;
}
```

Output on monitor:

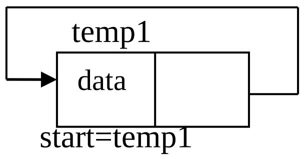
```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter a item to be inserted(F/M/L:=5
Created list after first insertion:=  5  10  20  30  40
```

Last Insertion (newitem=45):

If start= =NULL that means no linked list:

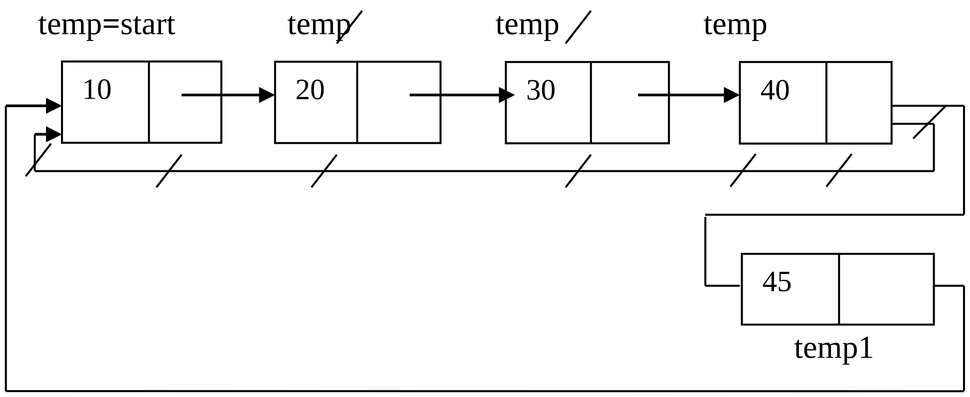


(Need to create a node named 'temp1'and later assigned it to start)



```
if(start= =NULL)
{
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp1->next=temp1;
start=temp1;
}
```

If there is a linked list; First insertion (newitem=5)



```
temp=start;
while(temp->next!=start)
{
temp=temp->next;
}
temp1=(node*)malloc(sizeof(node));
temp1->data=newitem;
temp->next=temp1;
temp1->next=start;
temp=temp1;
```

Display of circular linked list:

```
temp1=start;
if (temp1!=NULL)       // when no linked lists
{
  do
   {
    printf("  %d",temp1->data);
     temp1=temp1->next;
    } while (temp1! =start);
 }
```

Write a program in C to insert last new item into circular linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
	int data;
	struct list *next;
};
typedef struct list node;
node *insertion(node *start,int newitem);
void display(node *start);
int main()
{
	node *start,*prev,*temp;
	char ans;
	int newitem;
	start=NULL;
	do
	{
		fflush(stdout);
		printf("\n Do you want to create a node(Y/N):=");
		fflush(stdin);
		ans=toupper(getchar());
		if(ans=='Y')
		{
			if(start==NULL)
			{
				start=(node*)malloc(sizeof(node));
				fflush(stdout);
				printf("\n Enter a data:=");
				fflush(stdin);
				scanf("%d",&start->data);
				start->next=start;
				prev=start;
			}
			else
			{
				temp=(node*)malloc(sizeof(node));
				fflush(stdout);
				printf("\n Enter a data:=");
				fflush(stdin);
				scanf("%d",&temp->data);
				temp->next=start;
				prev->next=temp;
```

```
                                    prev=temp;
                            }
                    }
            }
            while(ans=='Y');
            printf("\n Created list:=");
           display(start);
            printf("\n");
            do{
            printf("\n Enter a item to be inserted(F/M/L:=");
            scanf("%d",&newitem);
           start=insertion(start,newitem);
            printf("\n Created list after first insertion:=");
            display(start);
            printf("\n");
            }while(1);
return 0;
}
node *insertion(node *start,int newitem)
{
   node *temp1,*temp;
            if(start==NULL)
            {
            temp1=(node*)malloc(sizeof(node));
           temp1->data=newitem;
           temp1->next=temp1;
            start=temp1;
            }
            else
            {
            temp=start;
            while(temp->next!=start)
            {
            temp=temp->next;
            }
            temp1=(node*)malloc(sizeof(node));
            temp1->data=newitem;
            temp->next=temp1;
            temp1->next=start;
            temp=temp1;
            }
return start;
}
void display(node *start)
{
            node *temp1;
            temp1=start;
            if(temp1!=NULL)
            {
            do
                        {
                        printf("  %d",temp1->data);
                     temp1=temp1->next;
                        }while(temp1!=start);
            }
                        return;
}
```

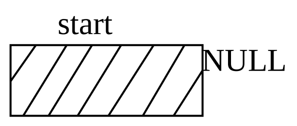
Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter a item to be inserted(F/M/L:=45
Created list after first insertion:=  10  20  30  40  45
```

<u>First and Last Deletion</u>

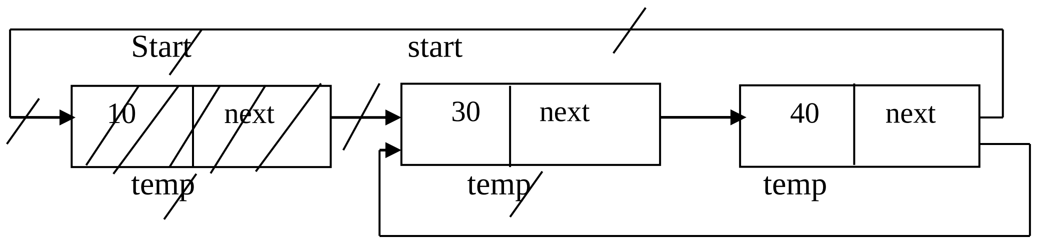
<u>First deletion (item = =10)</u>

<u>(No node that means start= =NULL)</u>



```
if(start= =NULL)
            {
             printf("\nNo linked list!");
            }
```

If there is a linked list:



```
temp=start;
  while(temp->next!=start)
  {
 temp=temp->next;
  }
     if(start->data= =item)
       {
       temp->next=start->next;
       temp1=start;
       start=temp->next;
       free(temp1);
     }
```

Write a program in C to delete first item from circular linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
        int data;
        struct list *next;
};
typedef struct list node;
node *deletion(node *start,int item);
void display(node *start);
int main( )
{
        node *start,*prev,*temp;
        char ans;
        int item;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
```

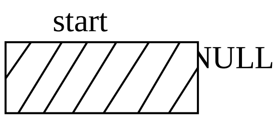
```
                    ans=toupper(getchar());
                    if(ans=='Y')
                    {
                            if(start==NULL)
                            {
                                    start=(node*)malloc(sizeof(node));
                                    fflush(stdout);
                                    printf("\n Enter a data:=");
                                    fflush(stdin);
                                    scanf("%d",&start->data);
                                    start->next=start;
                                    prev=start;
                            }
                            else
                            {
                                    temp=(node*)malloc(sizeof(node));
                                    fflush(stdout);
                                    printf("\n Enter a data:=");
                                    fflush(stdin);
                                    scanf("%d",&temp->data);
                                    temp->next=start;
                                    prev->next=temp;
                                    prev=temp;

                            }
                    }
            }
            while(ans=='Y');
            printf("\n Created list:=");
           display(start);
            printf("\n");
   do{
            printf("\n Enter a item to be deleted(F/M/L:=");
            scanf("%d",&item);
           start=deletion(start,item);
            printf("\n Created list after first deletion:=");
            display(start);
            printf("\n");
            }while(1);
            return 0;
}
node *deletion(node *start,int item)
{
   node *temp,*temp1;
            if(start==NULL)
            {
             printf("\nNo linked list!");
            }
            else
            {
            temp=start;
            while(temp->next!=start)
            {
           temp=temp->next;
            }
            if(start->data==item)
            {
            temp->next=start->next;
            temp1=start;
            start=temp->next;
            free(temp1);
    }
            }
return start;
}
void display(node *start)
{
            node *temp1;
            temp1=start;
    if(start!=NULL)
            {
            do
                        {
                        printf("  %d",temp1->data);
                     temp1=temp1->next;
                        }while(temp1!=start);
            }
                        return;
}
```

Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter a item to be deleted(F/M/L:=10
Created list after first deletion:=  20  30  40
```

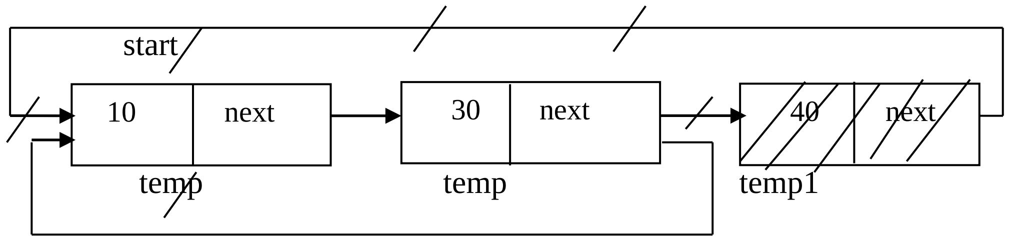
Last deletion (item = =40)

(No node that means start= =NULL)



```
if(start= =NULL)
            {
             printf("\nNo linked list!");
            }
```

If there is a linked list:



```
temp=start;
  while(temp->next->next!=start)
  {
  temp=temp->next;
  }
      if(temp->next->data= =item)
       {
       temp1=temp->next;
       temp->next=start;
       free(temp1);
        }
```

Write a program in C to delete first item from circular linked list:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
struct list
{
            int data;
            struct list *next;
```

```
};
typedef struct list node;
node *deletion(node *start,int item);
void display(node *start);
int main(){
        node *start,*prev,*temp;
        char ans;
        int item;
        start=NULL;
        do
        {
                fflush(stdout);
                printf("\n Do you want to create a node(Y/N):=");
                fflush(stdin);
                ans=toupper(getchar());
                if(ans=='Y')
                {
                        if(start==NULL)
                        {
                                start=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&start->data);
                                start->next=start;
                                prev=start;

                        }
                        else
                        {
                                temp=(node*)malloc(sizeof(node));
                                fflush(stdout);
                                printf("\n Enter a data:=");
                                fflush(stdin);
                                scanf("%d",&temp->data);
                                temp->next=start;
                                prev->next=temp;
                                prev=temp;

                        }
                }
        }
        while(ans=='Y');
        printf("\n Created list:=");
        display(start);
        printf("\n");
        do{
        printf("\n Enter a item to be deleted(F/M/L:=");
        scanf("%d",&item);
         start=deletion(start,item);
        printf("\n Created list after last deletion:=");
```

```
            display(start);
            printf("\n");
            }while(1);
            return 0;
}
node *deletion(node *start,int item)
{
    node *temp,*temp1;
            if(start==NULL)
             {
              printf("\nNo linked list!");
             }
     else
       {
            temp=start;
            while(temp->next->next!=start)
            {
            temp=temp->next;
            }
            if(temp->next->data==item)
            {
            temp1=temp->next;
            temp->next=start;
            free(temp1);
            }
        }
return start;
}
void display(node *start){
            node *temp1;
            temp1=start;
            if(start!=NULL)
            {
            do
                        {
                        printf("  %d",temp1->data);
                        temp1=temp1->next;
                        }while(temp1!=start);
            }
                        return;
}
```

Output on monitor:

```
Do you want to create a node(Y/N):=y
Enter a data:=10
Do you want to create a node(Y/N):=y
Enter a data:=20
Do you want to create a node(Y/N):=y
Enter a data:=30
Do you want to create a node(Y/N):=y
Enter a data:=40
Do you want to create a node(Y/N):=n
Created list:=  10  20  30  40
Enter a item to be deleted(F/M/L:=40
Created list after last deletion:=  10  20  30
```