Linked list

A **Linked list** is a collection of nodes where each node contains data and link field. Data field contains data and link field contains the address of next node.

A linked list is a non-sequential collection of data items. It is a dynamic data structure. For every data item in a linked list, there is an associated pointer that would give the memory location of the next data item in the linked list. The data items in the linked list are not in consecutive memory locations. They may be anywhere, but the accessing of these data items is easier as each data item contains the address of the next data item.

DATA	LINK

Figure 1: Representation of node

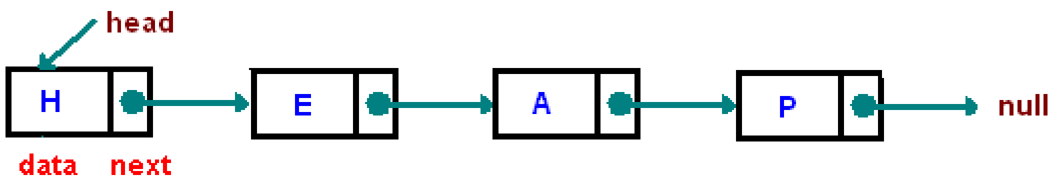
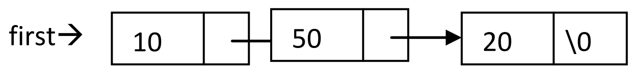


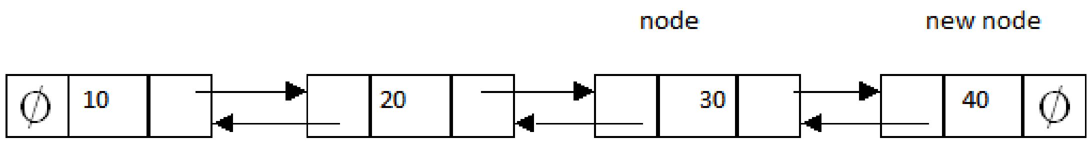
Figure 2: Example of Linked list

Basically we can put linked lists into the following four items:

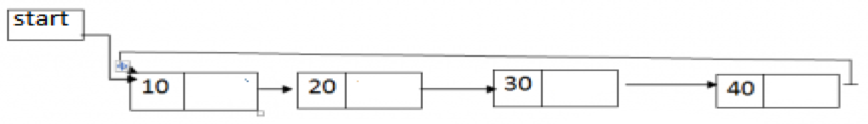
1. **Single Linked List.**



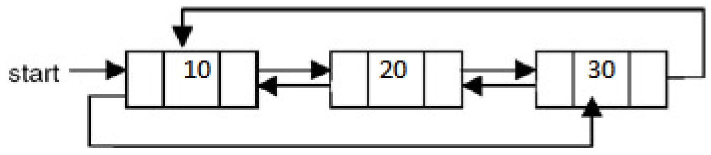
2. Double Linked List.



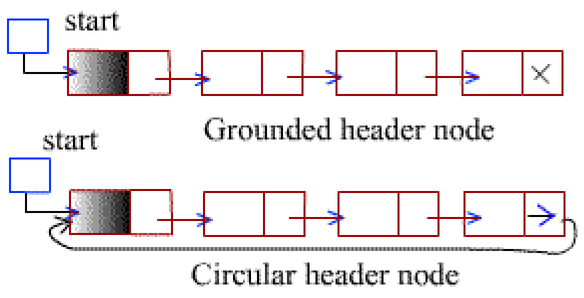
3. Circular Linked List.



4. Circular Double Linked List.



5 A header linked list is a linked list which always contains a special node called the ***header node*** at the beginning of the list.



7. Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, Sem, PhNo

a. Create a SLL of N Students Data by using front insertion

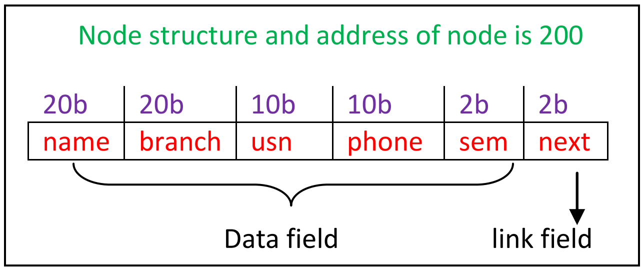
b. Display the status of SLL and count the number of nodes in it

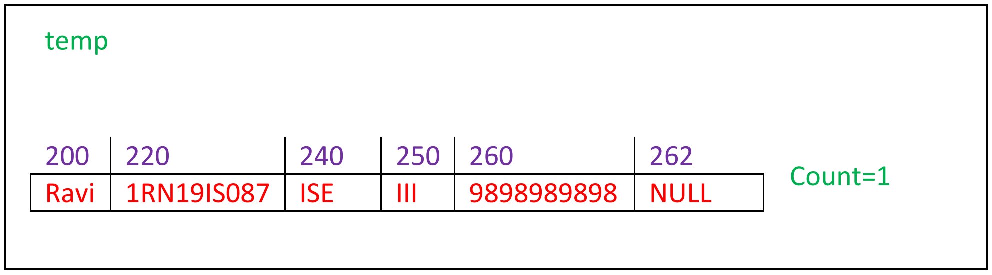
c. Perform Insertion / Deletion at End of SLL

d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)

e. Exit

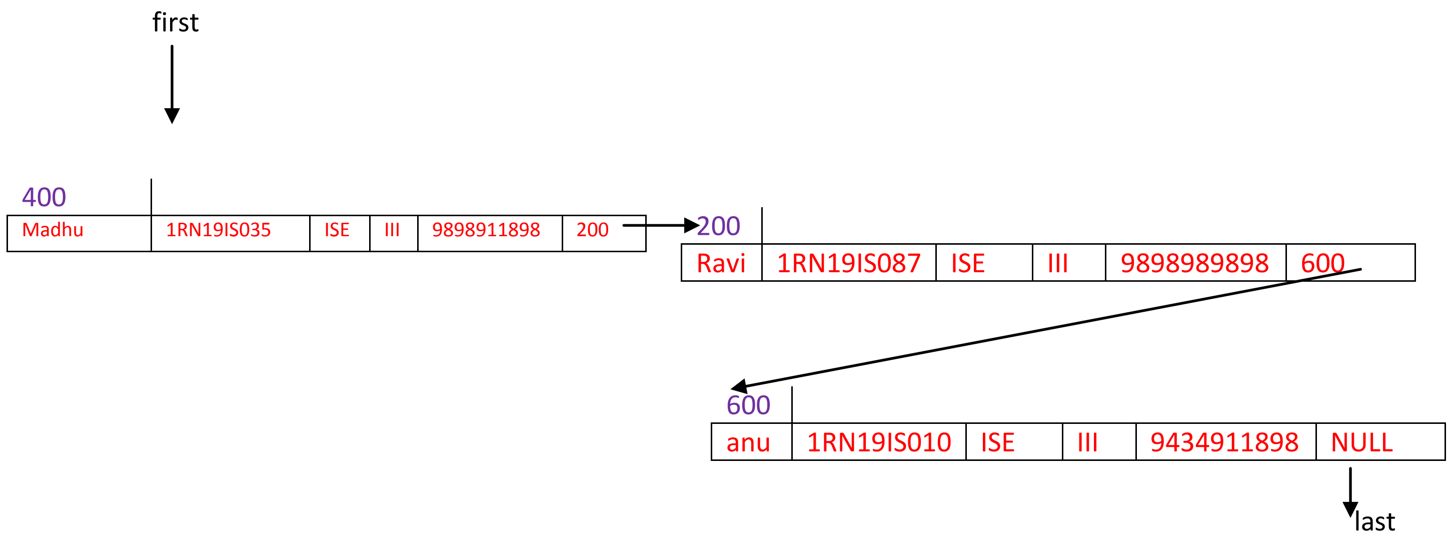
```c
#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<stdlib.h>

int count=0; // counter to count number of nodes in linked list

//structure to define a node in SLL

struct node

{

            char name[20],branch[20],usn[10],phone[10];

             int sem;

            struct node *next;

}*first=NULL,*last=NULL,*temp=NULL,*temp1=NULL;

//Create function creates a node in linked list

void create()

{

            int sem,phno;

            char name[20],usn[10],branch[20];

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

            printf("Enter the student details \n");

            printf("\nName, USN, Branch,Sem,Phone Number :");

           scanf("%s%s%s%d%s",temp->name,temp->usn,temp->branch,temp->sem,temp->phno);

            temp->next=NULL;

            count++;

}
```

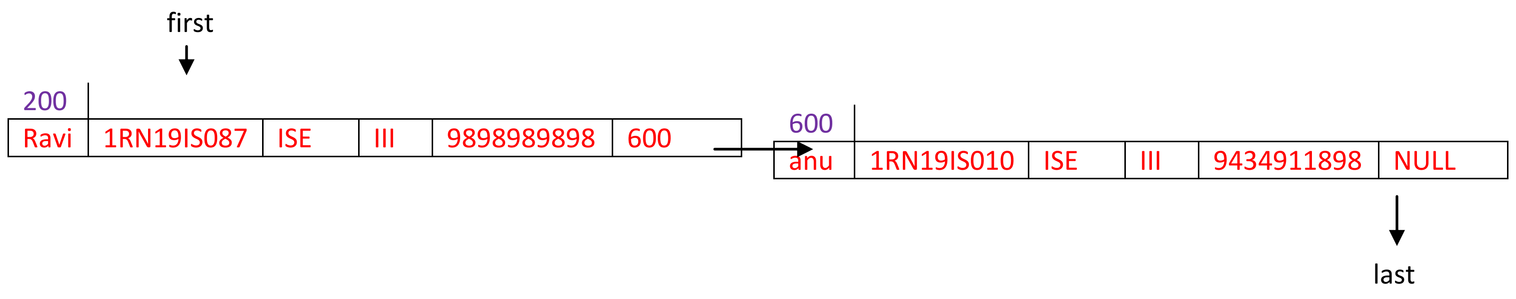




```
void deletefront()
{
        temp=first;
        if(first==NULL) // initially first points to NULL, We say list is empty and return
        {
                printf("\n list is empty");
                return;
        }
        if(temp->next==NULL)
// IF only one node exists in list , print the details and delete the node using free() and assign first to null
        {
                printf("The deleted node is \n");
                printf("%s\t%s\t%s\t%d\t%s",temp->name,temp->usn,temp->branch,temp->sem,temp-
>phno);
                free(temp);
                first=NULL;
        }
        else  // its more then more node condition and delete first
        {
                first=temp->next;
                printf("The deleted node is \n");
                printf("%s\t%s\t%s\t%d\t%s",temp->name,temp->usn,temp->branch,temp->sem,temp-
>phno);
                free(temp);
        }
        count--;
}
```

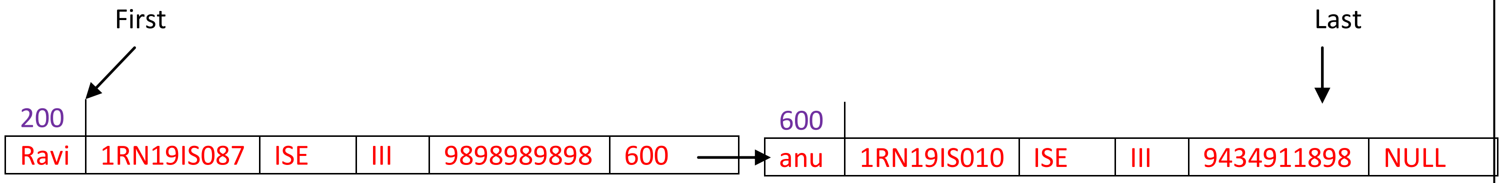
```
void deletefront()

{

        temp=first; //temp=400

        if(first==NULL) //400==NULL→FALSE

        if(temp->next==NULL) //200==NULL→FALSE

        else  // its more then one node condition and delete first

        {

                first=temp->next; // first=200

                printf("The deleted node is \n");

                printf("%s\t%s\t%s\t%d\t%s",temp->name,temp->usn,temp->branch,temp->sem,temp-
>phno); // madhu 1RN19IS035 ISE III 9898911898

                free(temp); //free(200)


        }

        count--;    // count=2

}
```

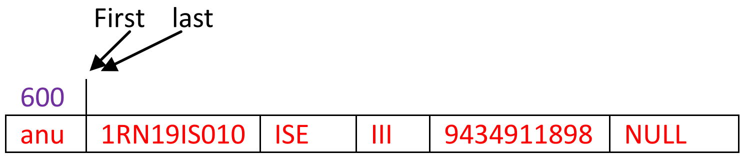




```
void deleteatend()
{
        temp=first;
        if(first==NULL) // initially first points to NULL, We say list is empty and return
        {
                 printf("\n list is empty");
                 return;
        }
        if(temp->next==NULL)
// IF only one node exists in list , print the details and delete the node using free() and assign first to null
        {
                printf("The deleted node is \n");
                printf("%s\t%s\t%s\t%d\t%s",temp->name,temp->usn,temp->branch,temp->sem,temp->phno);
                free(temp);
                first=NULL;
        }
        else // its more then one node condition and traverse till last node and delete last node
        {
                while(temp->next!=last)
                        temp=temp->next;
                printf("The deleted node is \n");
                printf("%s\t%s\t%s\t%d\t%s",last->name,last->usn,last->branch,last->sem,last->phno);
                free(last);
                last=temp;
                last->next=NULL;
        }
        count--;

}
```

```
void deleteatend()
{
        temp=first; //temp=200
        if(first==NULL) //200==NULL→FALSE
        if(temp->next==NULL) //600==NULL→FALSE
        else  // its more then one node condition and delete first
        {
        while(temp->next!=last) // 200!=200(while doesnot work as we have only 2 nodes
                        temp = temp->next;
                printf("The deleted node is \n");
                printf("%s\t%s\t%s\t%d\t%s",last->name,last->usn,last->branch,last->sem,last->phno);
                    anu     1RN19IS010    ISE       III      9434911898
                free(last); //last node is deleted using free()
                last=temp; last=200
                last->next=NULL; //last->next=NULL

        }
        count--;    // count=1
}
```



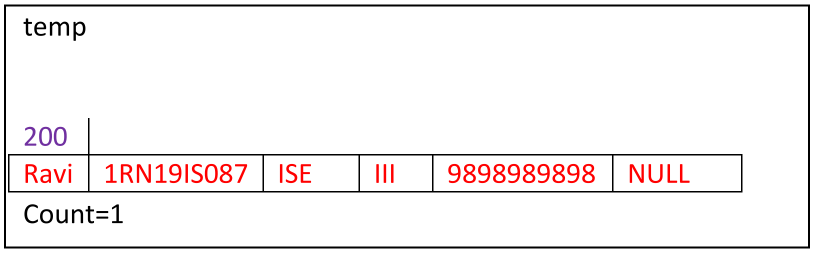


```
void insertatfirst() // function to insert at first

{

            create();

            if(first==NULL)

            {

                        first=temp;

                        last=first;

            }

            else

            {

                        temp->next=first;

                        first=temp;

            }

}
```

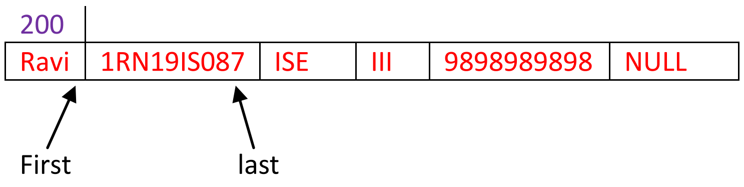
I iteration of insertatfirst()

*first=NULL//initially first is pointer to structure which is pointing to null as linked list is empty

When you call create function its create a node called temp



```
if(first==NULL)
(NULL==NULL)
            {
                        first=temp; first=200
                        last=first; last=200
            }
```

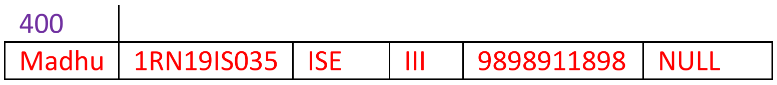


II iteration of insertatfirst()

first=200

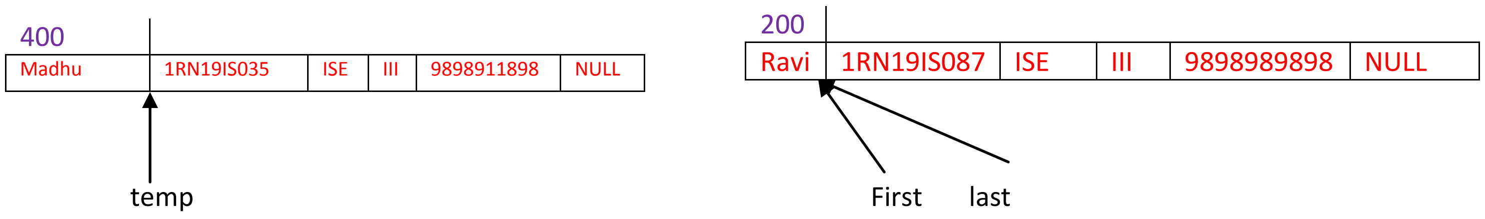
When you call create function its create a node called temp

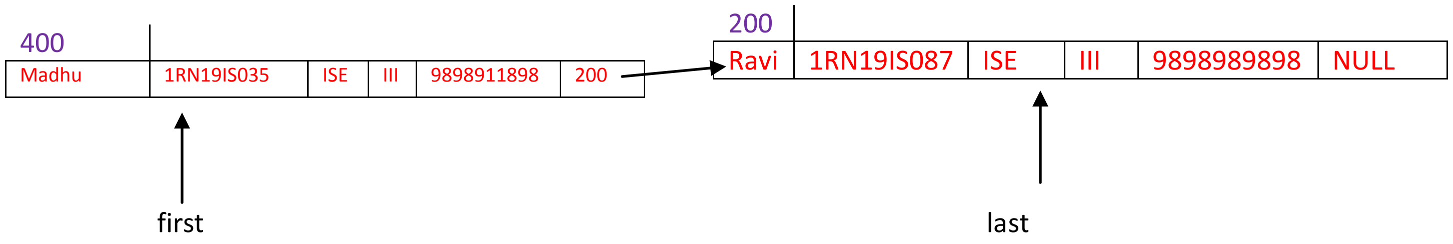
temp



Count=2

```
if(first==NULL)
(200==NULL)→false

else

        {

                temp->next=first; // temp->next=200

                first=temp;//first=400

        }
```





```
void insertatlast()
{
        create();
        if(first==NULL)
        {
                first=temp;
                last=first;
        }
        else
        {
        last->next=temp;
        last=temp;
        }
}
```

I iteration of insertatlast **() if its called first in main() program menu**

*first=NULL//initially first is pointer to structure which is pointing to null as linked list is empty

When you call create function its create a node called temp

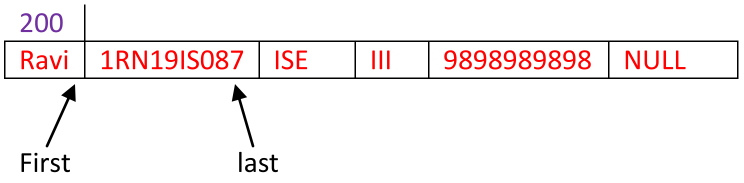
temp

200

Ravi	1RN19IS087	ISE	III	9898989898	NULL

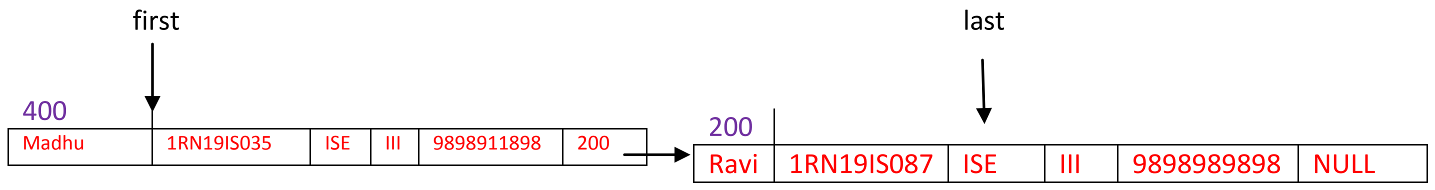
Count=1

```
if(first==NULL)
(NULL==NULL)
        {
                first=temp; first=200
                last=first; last=200
        }
```



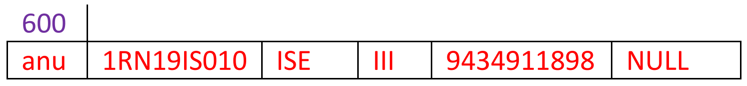
<u>I iteration of insertatfirst() after insertfirst()is done</u>

first=400


first
last
400
Madhu | 1RN19IS035 | ISE | III | 9898911898 | 200
200
Ravi | 1RN19IS087 | ISE | III | 9898989898 | NULL

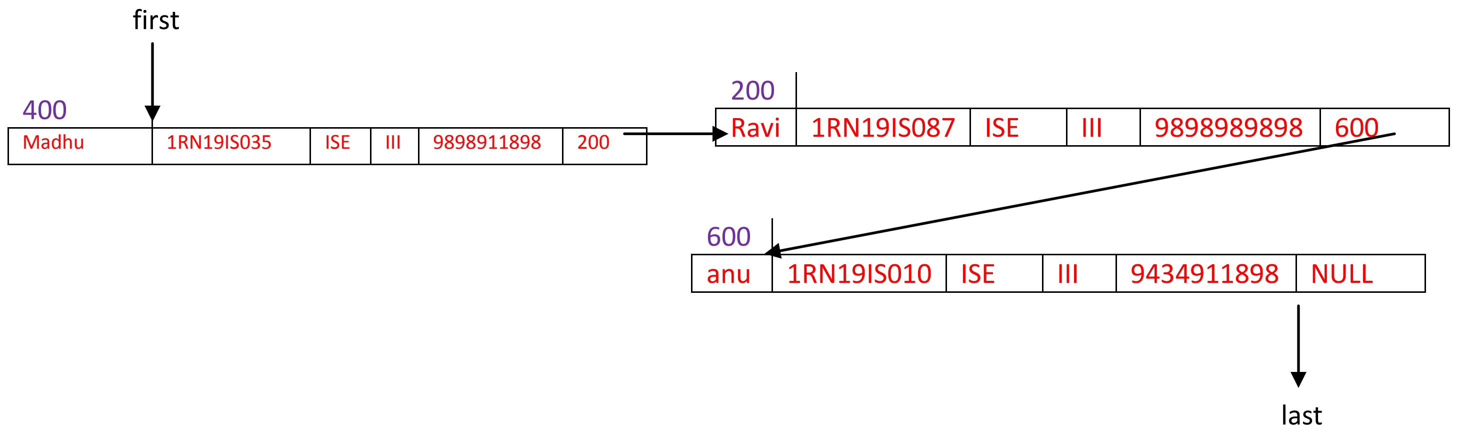
When you call create function its create a node called temp

temp

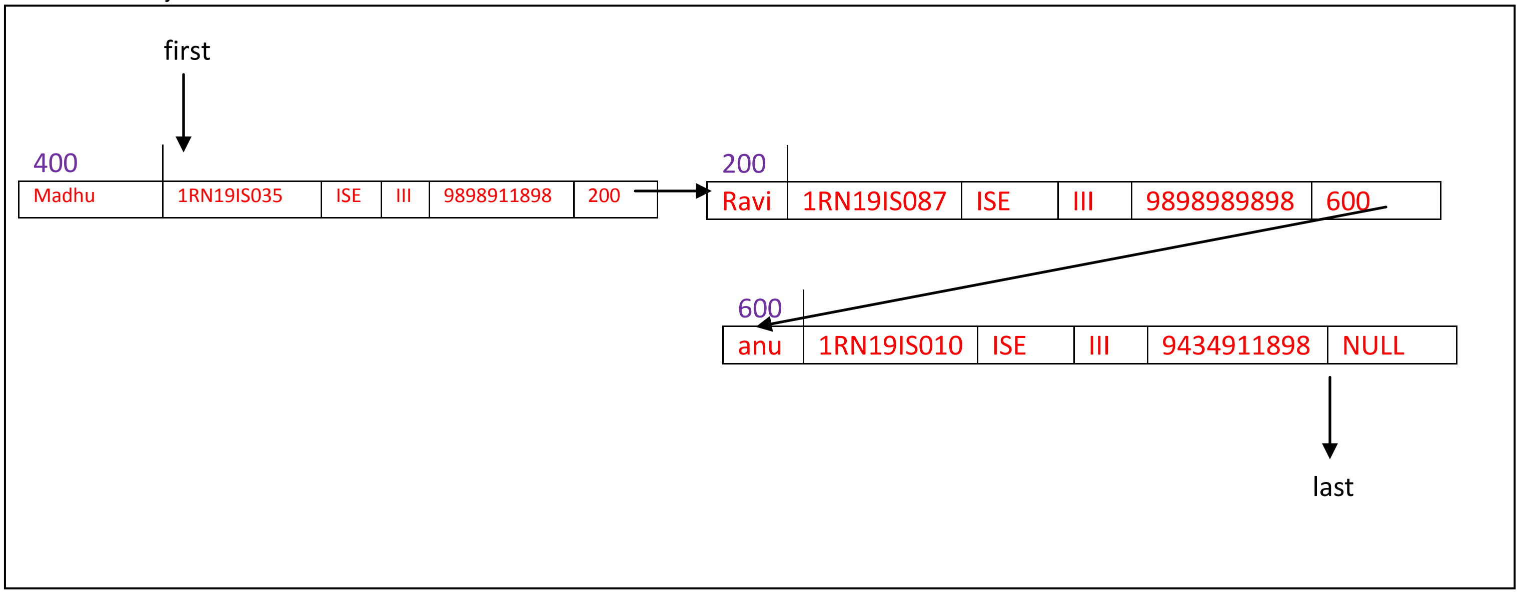


Count=3

```
if(first==NULL)
(400==NULL)→false
else
        {
                last->next=temp; //last->next=600
                last=temp; //last=600
        }
```


first
400
Madhu | 1RN19IS035 | ISE | III | 9898911898 | 200
200
Ravi | 1RN19IS087 | ISE | III | 9898989898 | 600
600
anu | 1RN19IS010 | ISE | III | 9434911898 | NULL
last

```
void display()
{
        if(first==NULL)
        {
                printf("\n list is empty");
        }
        else
        {
                temp=first;
                printf("The node is \n");
                while(temp!=NULL)
                {

                        printf("%s\t%s\t%s\t%d\t%s--->",temp->name,temp->usn,temp->branch,temp-
>sem,temp->phno);
                        temp=temp->next;
                        //printf("\n");
                }
        }
}
```



```
else
        {
                temp=first; //temp=400
                printf("The node is \n");
                while(temp!=NULL) // 400!=NULL
                {
                          Madhu       1RN19IS035     ISE    III    9898911898        →
                          temp=200;
                }
               while(temp!=NULL) // 200!=NULL
                {
                      Ravi   1RN19IS087   ISE     III     9898989898    →
                        temp=600;
                 }
                while(temp!=NULL) // 600!=NULL
                {
                    anu           1RN19IS010     ISE    III    9434911898       →
                          temp=NULL;
                }
               while(temp!=NULL) // NULL!=NULL
                //
        }
}
```

```
void main()
{
        int ch,i,n;
        clrscr();
        while(1)
        {
                printf("\n1.Insert n details student ");
                printf("\n2.Insert at beginning");
                printf("\n3.Insert at last");
                printf("\n4.Delete from begining");
                printf("\n5.Delete from last");
                printf("\n6.Display");
                printf("\n7.Exit");
                printf("\nEneter your choice : ");
                scanf("%d",&ch);
                switch(ch)
                {
                 case 1 : printf("\nEnter the value of n ");
                      scanf("%d",&n);   //n=2
                      for(i=0;i<n;i++)// loop executes twice
                        insertatfirst();
                      break;
                 case 2 : insertatfirst();
                      break;
                 case 3 : insertatlast();
                      break;
                 case 4 : deletefront();
                      break;
                 case 5 : deleteatend();
```

```
                                        break;
                        case 6 : display();
                                        break;
                        case 7 : exit(1);
                        default: printf("\n Wrong Input, try again");
                       }
               }
       }
```