Deleting a node in SLL

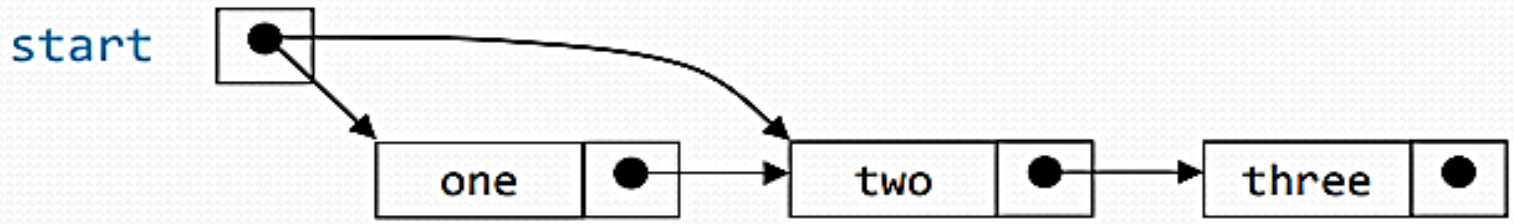
Here also we have three cases:-

- Deleting the first node
- Deleting the last node
- Deleting the intermediate node

Deleting the first node

Here we apply 2 steps:-

- Making the start pointer point towards the 2nd node
- Deleting the first node using delete keyword

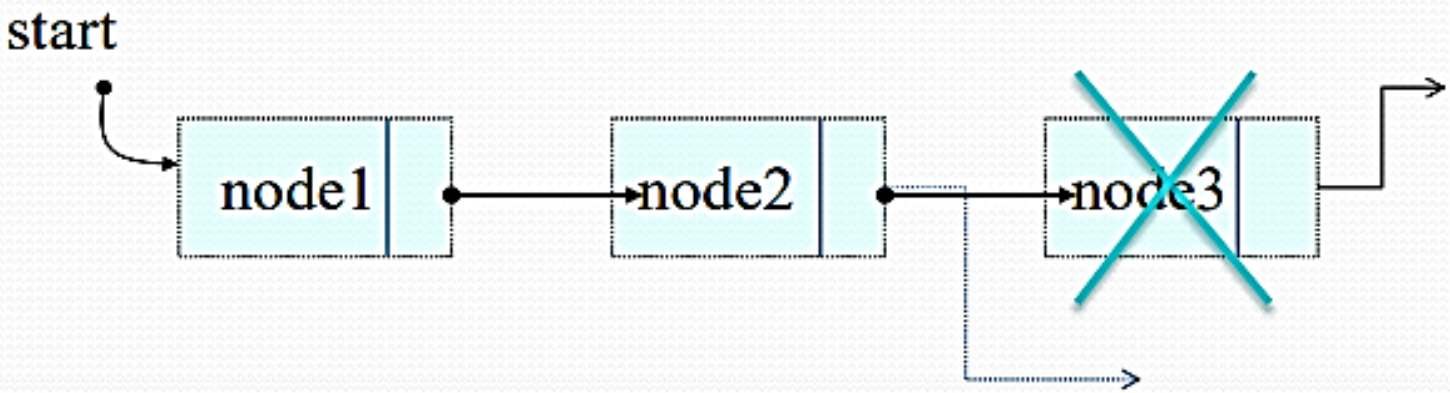


```
void del_first()
{
      if(start==NULL)
     cout<<"\nError.....List is empty\n";
     else
     {
       node* temp=start;
       start=temp->link;
       delete temp;
cout<<"\nFirst node deleted successfully....!!!";
     }
}
```

Deleting the last node

Here we apply 2 steps:-

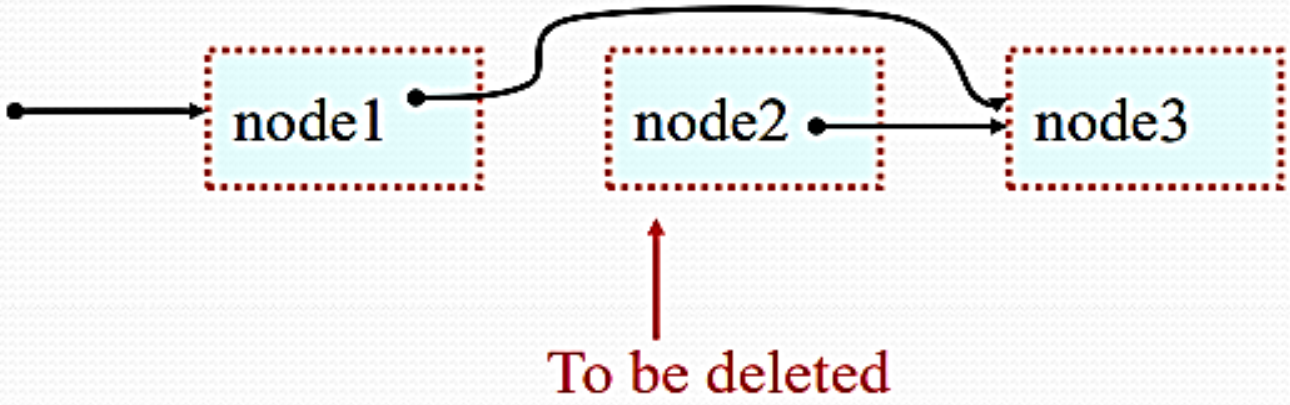
- Making the second last node's next pointer point to NULL
- Deleting the last node via delete keyword



```
void del_last()
{
        if(start==NULL)
        cout<<"\nError....List is empty";
        else
                {
                   node* q=start;
                           while(q->link->link!=NULL)
                            q=q->link;
                   node* temp=q->link;
                   q->link=NULL;
                   delete temp;
                   cout<<"\nDeleted successfully...";
                }
        }
}
```

Deleting a particular node

Here we make the next pointer of the node previous to the node being deleted ,point to the successor node of the node to be deleted and then delete the node using delete keyword



```
void del(int c)
{
node* q=start;
      for(int i=2;i<c;i++)
      {
       q=q->link;
             if(q==NULL)
             cout<<"\nNode not found\n";
      }
      if(i==c)
      {
       node* p=q->link;          //node to be deleted
      q->link=p->link;       //disconnecting the node p
      delete p;
      cout<<"Deleted Successfully";
      }
}
```

Searching a SLL

- Searching involves finding the required element in the list
- We can use various techniques of searching like linear search or binary search where binary search is more efficient in case of Arrays
- But in case of linked list since random access is not available it would become complex to do binary search in it
- We can perform simple linear search traversal

In linear search each node is traversed till the data in the node matches with the required value

```
void search(int x)
{
  node*temp=start;
    while(temp!=NULL)
      {
        if(temp->data==x)
         {
           cout<<"FOUND "<<temp->data;
           break;
         }
       temp=temp->next;
      }
}
```

Traversing a linked list

```
   struct Node {
      Object element;
      Node *next;
   };

Node *pWalker;
int count = 0;

cout <<"List contains:\n";

for (pWalker=pHead; pWalker!=NULL;
     pWalker = pWalker->next)
{
    count ++;
    cout << pWalker->element << endl;
}
```