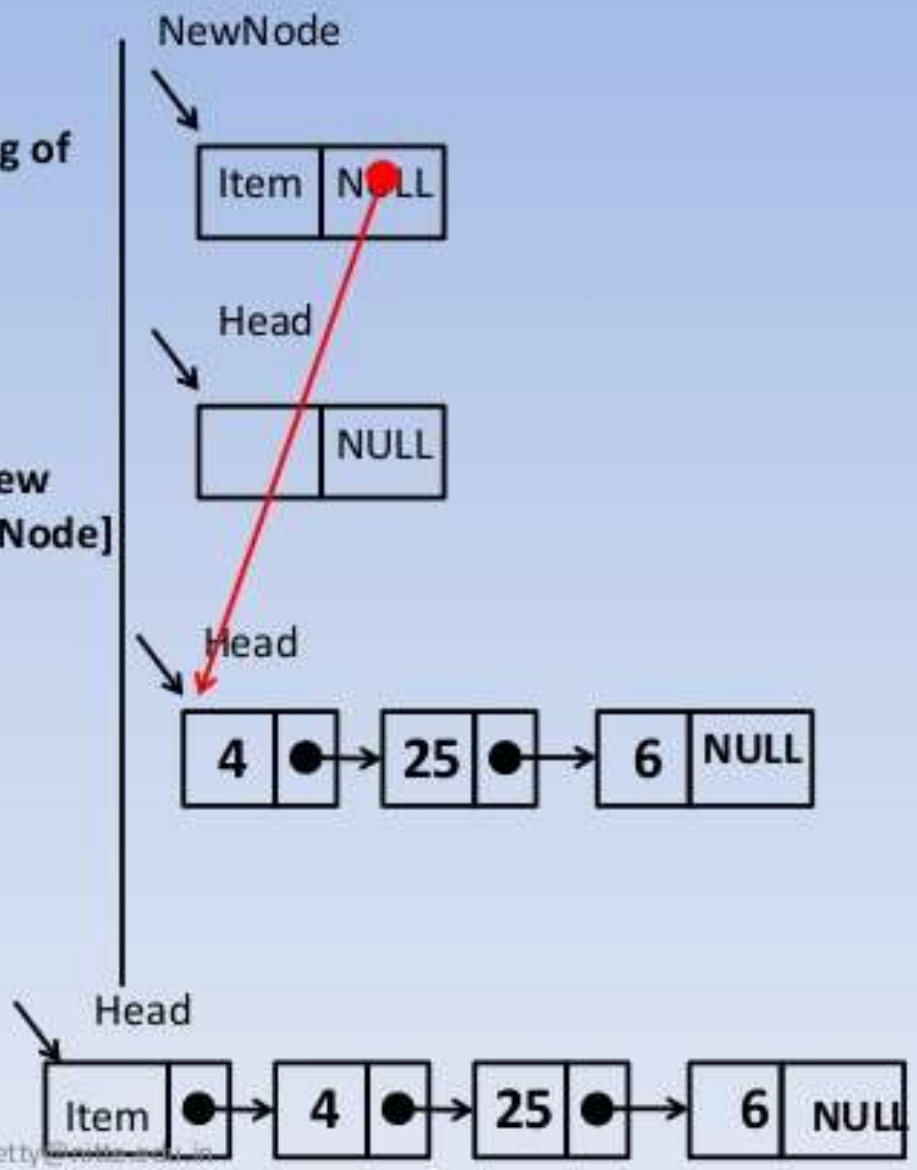
Algorithm Link-List

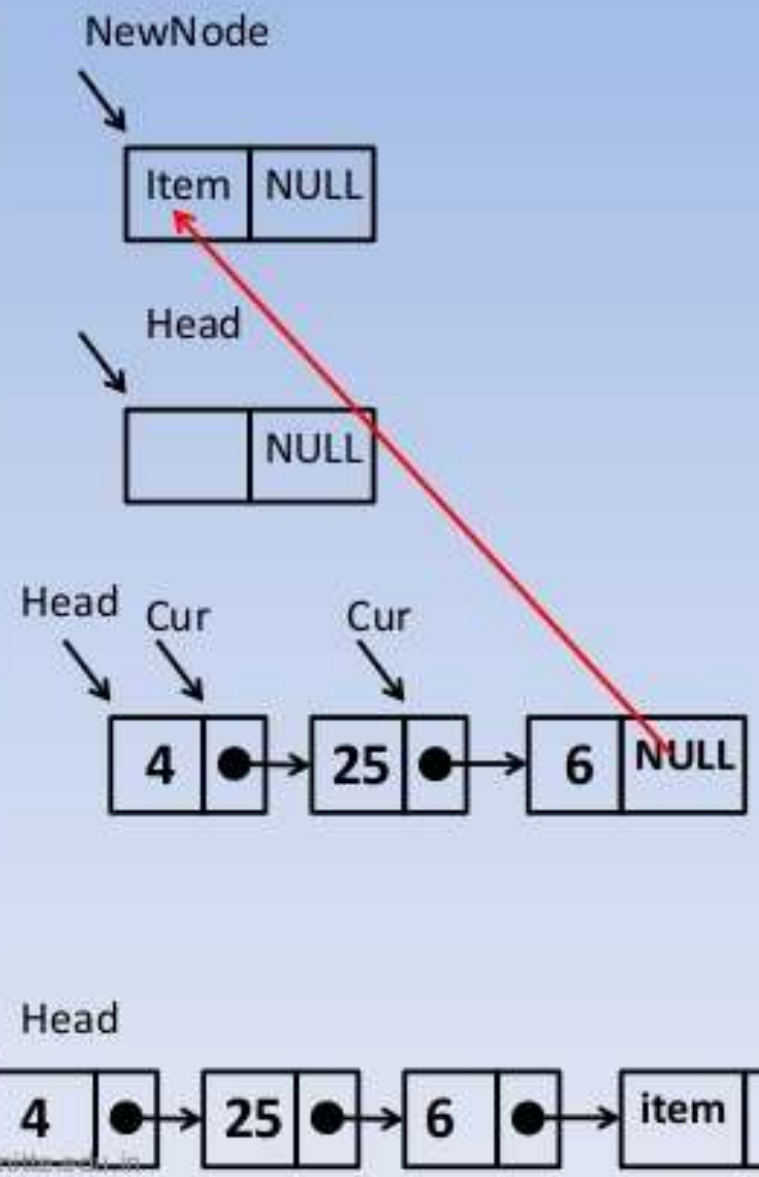
1) Insert a node at the beginning of the list

```
// Algorithm : Insertion (Item)
// Description : Inserts a node at the beginning of
the SLL
Step1: Start
Step 2: NewNode=getnode()
        NewNode -> link = NULL
        NewNode -> info = Item
Step 3: [ If the Linked list is empty, then the New
          Node Created is a first node or Head Node]
        if(Head==NULL)
          Head=NewNode
          return
        End if
Step 4: [If the linked list is not empty]
        NewNode -> link= Head
        Head= NewNode
        Free(NewNode)
Step 5: Return (Stop)
```



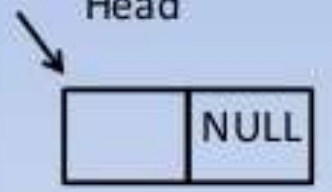
2) Insert a node at the End of the list

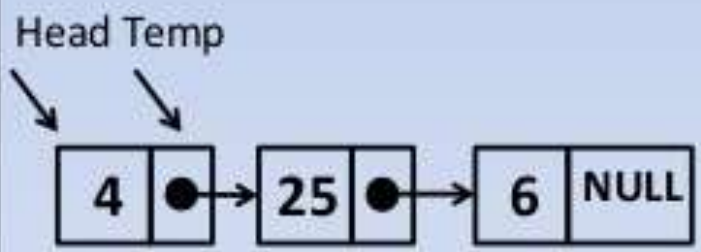
```
// Algorithm : Insertion (Item)
// Description : Inserts a node at the End of the SLL
Step1: Start
Step 2: NewNode=getnode()
        NewNode -> link = NULL
        NewNode -> info = Item
Step 3: [ If the Linked list is empty, then the New  Node Created is a first node or Head Node]
        if(Head==NULL)
          Head=NewNode
          return
        End if
Step 4: [If the linked list is not empty]
        Cur= Head
         While (Cur-> link !=NULL)
                Cur=Cur->link
         end while
Step 5:  Cur -> link= NewNode
Step 6: Free(NewNode)
        Free (Cur)
Step 5: Return (Stop)
```

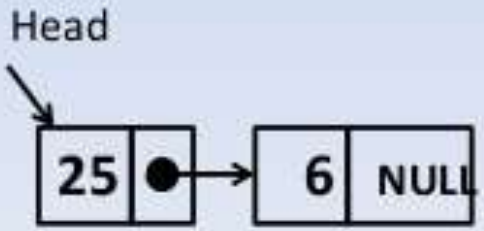


1) Delete a node from the beginning of the list

```
// Algorithm : Deletion (Item)
// Description : Delete a node from the beginning of the SLL
Step1: Start
Step 2: [If Empty List]
        If (Head==NULL)
         Display ("List Empty")
         Return
         EndIf
Step 3: [If the linked list is not empty]
        Temp=Head
         Head=Head-> link
         display (temp->info)
         Free(Temp)
Step 4: Return (Stop)
```



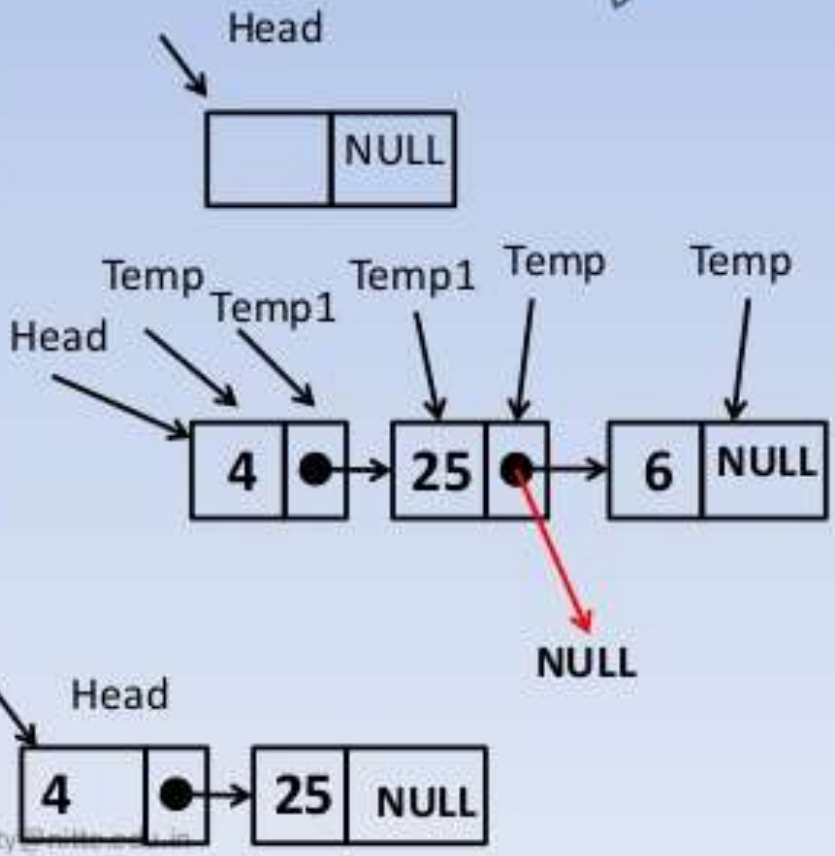




Shashankshetty@nitte.edu.in

2) Delete a node from the End of the list

```
// Algorithm : Deletion (Item)
// Description : Delete a node from the end of the
SLL
Step1: Start
Step 2: [If Empty List]
        If (Head==NULL)
         Display ("List Empty")
         Return
         EndIf
Step 3: [If the linked list is not empty]
        Temp=Head
         While(temp->link!=NULL)
                temp1=temp
                temp=temp->link
         end while
         temp1->next=NULL
        display (temp->info)
         Free(Temp)
Step 4: Return (Stop)
```



3) Insert a node at the middle of two nodes or at the designated position

```
// Algorithm : Insertion (Item, position)
// Description : Inserts a node at the middle or at the designated position of the SLL
Step1: Start
Step 2: NewNode=getnode()
        NewNode -> link = NULL
        NewNode -> info = Item
Step 3: [ If the Linked list is empty, then the New  Node Created is a first node or Head Node]
        if(Head==NULL)
          Head=NewNode
          return
        End if
Step 4: [If the linked list is not empty]
        Cur= Head
         While (Cur!=position-1)
                Cur=Cur->link
         end while
         Cur1=Cur->link
Step 5:  Cur->link= NewNode
         NewNode->link= Cur1
Step 6: Free(NewNode)
        Free (Cur)
        Free(Cur1)
Step 5: Return (Stop)
```