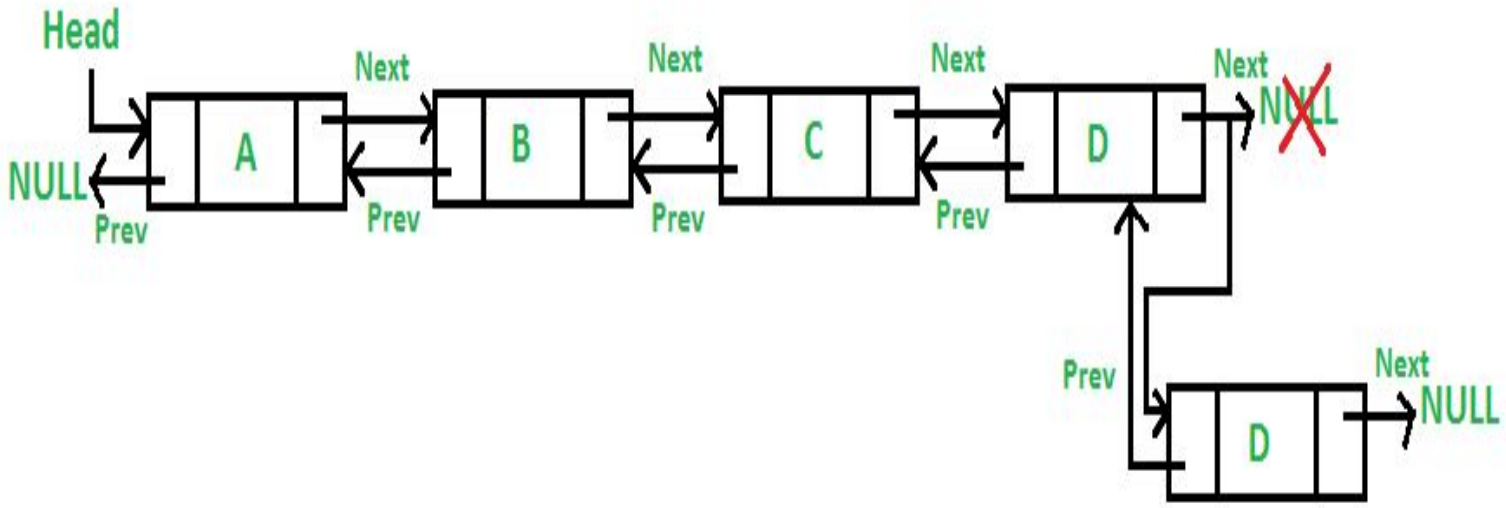
Insertion at the end of Doubly Linked List

- Insertion at the end of Doubly Linked List



Insertion at end in Doubly linked list

```
# A single node of a doubly linked list
class Node:
    def __init__(self, data):
        self.prev = None
        self.data = data
        self.next = None

# A Linked List class with a single head node
class DoublyLinkedList:
    def __init__(self):
        self.head = None
```

```
# Insertion method for the doubly linked list at end
    def insert_end(self, data):
        newNode = Node(data)
        if(self.head==None):
            self.head = newNode

        else:
            temp=self.head
            while(temp.next!=None):
                temp=temp.next

            temp.next=newNode
            newNode.prev=temp
```

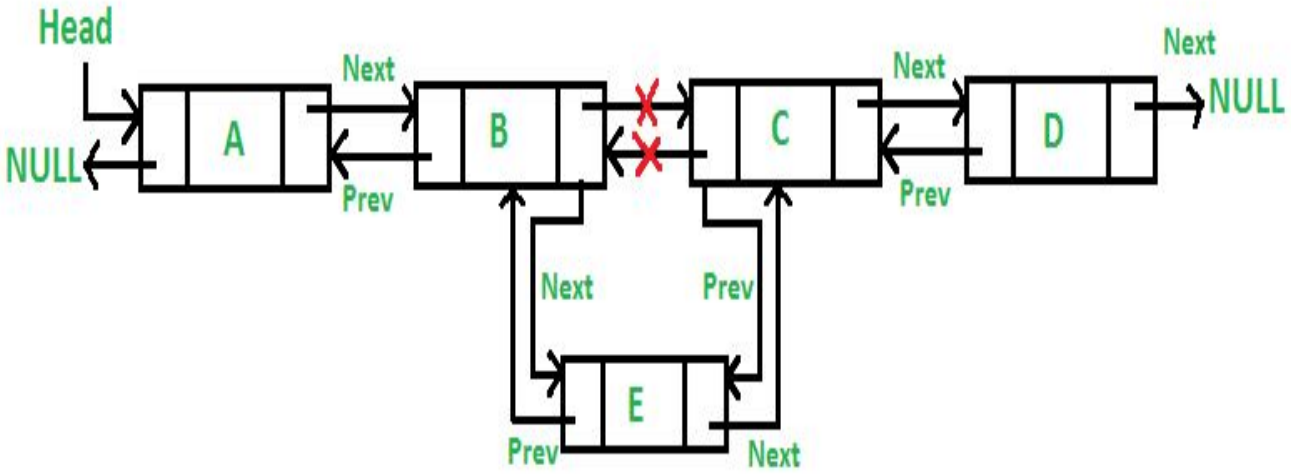
Insertion at end in Doubly linked list (contd..)

```
# print method for the linked list
    def printLL(self):
        current = self.head
        if(current!=None):
            print("The List Contains:",end="\n")
            while(current!=None):
                print(current.data)
                current = current.next
        else:
            print("List is Empty.")
```

```
# Singly Linked List with creation and
      print methods
LL = DoublyLinkedList()
LL.insert_end(3)
LL.insert_end(4)
LL.insert_end(5)
LL.insert_end(6)
LL.printLL()
```

Insertion in Doubly Linked List (at position)

- Insertion at given position in Doubly Linked List



Insertion in Doubly Linked List (at position)

```
# A single node of a doubly linked list
class Node:
    def __init__(self, data):
        self.prev = None
        self.data = data
        self.next = None


# A Linked List class with a single head
    node
class DoublyLinkedList:
    def __init__(self):
        self.head = None
```

```
# creation method for the doubly
    linked list
    def create(self, data):
        newNode = Node(data)
        if(self.head==None):
            self.head = newNode

        else:
            temp=self.head
            while(temp.next!=None):
                temp=temp.next

            temp.next=newNode
            newNode.prev=temp
```

Insertion in Doubly Linked List (at position) (contd..)

```
# insertion method for the doubly
   linked list at given position


   def insert_position(self, data, pos):
     newNode = Node(data)
     if(pos<1):
       print("\nPosition should be >=1.")

     elif(pos==1):
       newNode.next=self.head
       self.head.prev=newNode
       self.head=newNode
     else:
       current=self.head
       for i in range(1, pos-1):
          if(current!=None):
             current=current.next

       if(current!=None):
          newNode.next=current.next
          current.next.prev=newNode
          current.next=newNode
          newNode.prev=current

       else:
          print("\nThe previous node is null.")
```

Insertion in Doubly Linked List (at position) (contd..)

```
# print method for the linked list
    def printLL(self):
        current = self.head
        if(current!=None):
            print("The List Contains:",end="\n")
            while(current!=None):
                print(current.data)
                current = current.next
        else:
            print("List is Empty.")
```

```
# Singly Linked List with creation and
      print methods
LL = DoublyLinkedList()
LL.create(3)
LL.create(4)
LL.create(5)
LL.create(6)
LL.create(7)
LL.printLL()
LL.insert_position(4, 9)
LL.printLL()
```

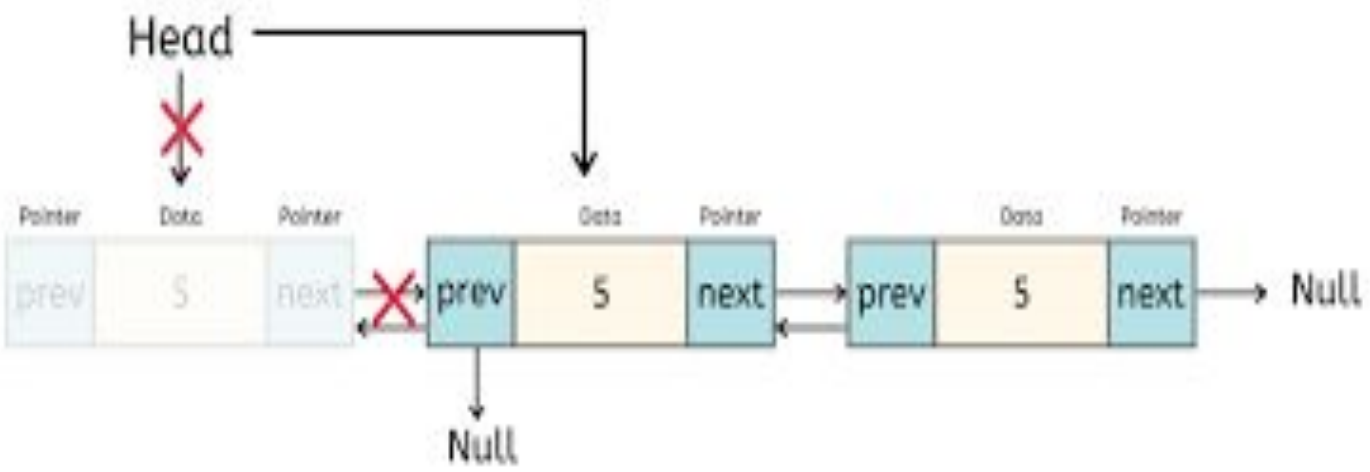
Deletion in a Doubly Linked List

- Similar to single linked list there are three possible positions where we can enter a new node in a doubly linked list –
  - **Deletion at beginning**
  - **Deletion at end**
  - **Deletion from given position**

- Deleting new node in linked list is a more than one step activity.

Deletion in Doubly Linked List (from beginning)

- **Deletion from beginning**



Deletion in Doubly Linked List (from beginning)

```
# A single node of a doubly linked list
class Node:
    def __init__(self, data):
        self.prev = None
        self.data = data
        self.next = None

# A Linked List class with a single head node
class DoublyLinkedList:
    def __init__(self):
        self.head = None
```

```
# creation method for the doubly linked list
    def create(self, data):
        newNode = Node(data)
        if(self.head==None):
            self.head = newNode

        else:
            temp=self.head
            while(temp.next!=None):
                temp=temp.next

            temp.next=newNode
            newNode.prev=temp
```

Deletion in Doubly Linked List (from beginning) (contd..)

```
#Delete first node of the list
  def del_beg(self):
    if(self.head == None):
      print("Underflow-Link List is empty")
    else:
      temp = self.head
      self.head = self.head.next
      self.head.prev=None
      print("the deleted element is", temp.data)
      temp = None
```

```
# print method for the linked list
  def printLL(self):
    current = self.head
    if(current!=None):
      print("The List
Contains:",end="\n")
      while(current!=None):
        print(current.data)
        current = current.next
    else:
      print("List is Empty.")
```

Deletion in Doubly Linked List (from beginning) (contd..)

```
# Doubly Linked List with creation, deletion and print methods
LL = DoublyLinkedList()
LL.create(3)
LL.create(4)
LL.create(5)
LL.create(6)
LL.printLL()
LL.del_beg()
LL.printLL()
```