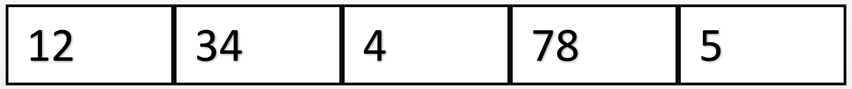
Linked Lists

Ways in which linked lists differ from arrays

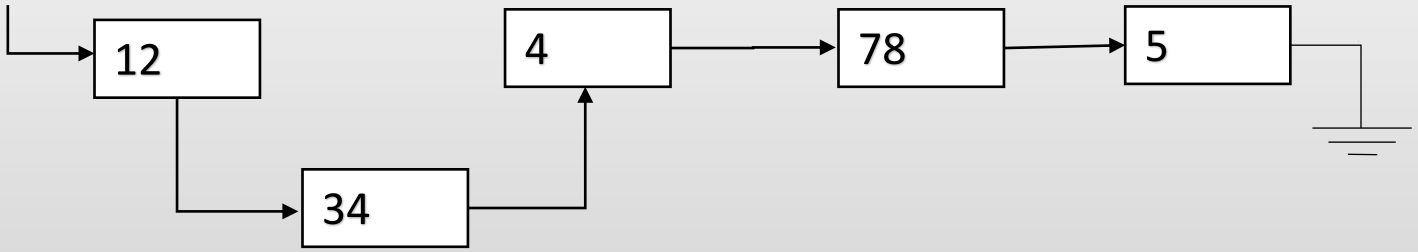
Array – each item occupies a particular position and can be directly accessed using an index number.

Linked list – need to follow along the chain of element to find a particular element. A data item cannot be accessed directly.

Array →



Linked List →



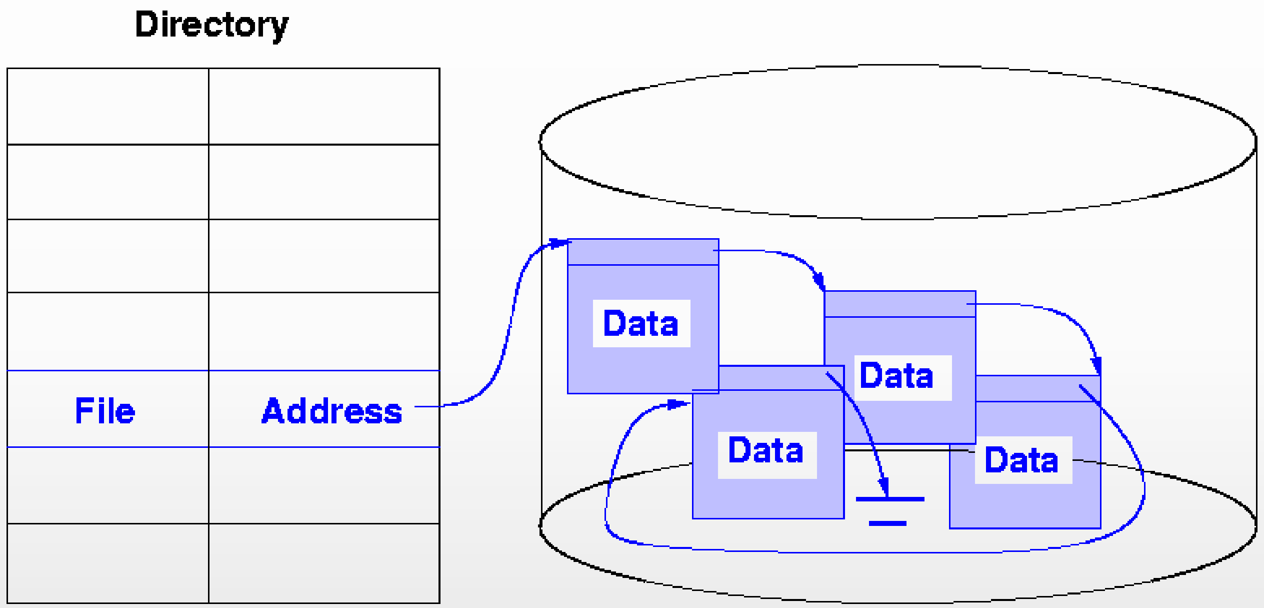
Applications of linked list in real world-

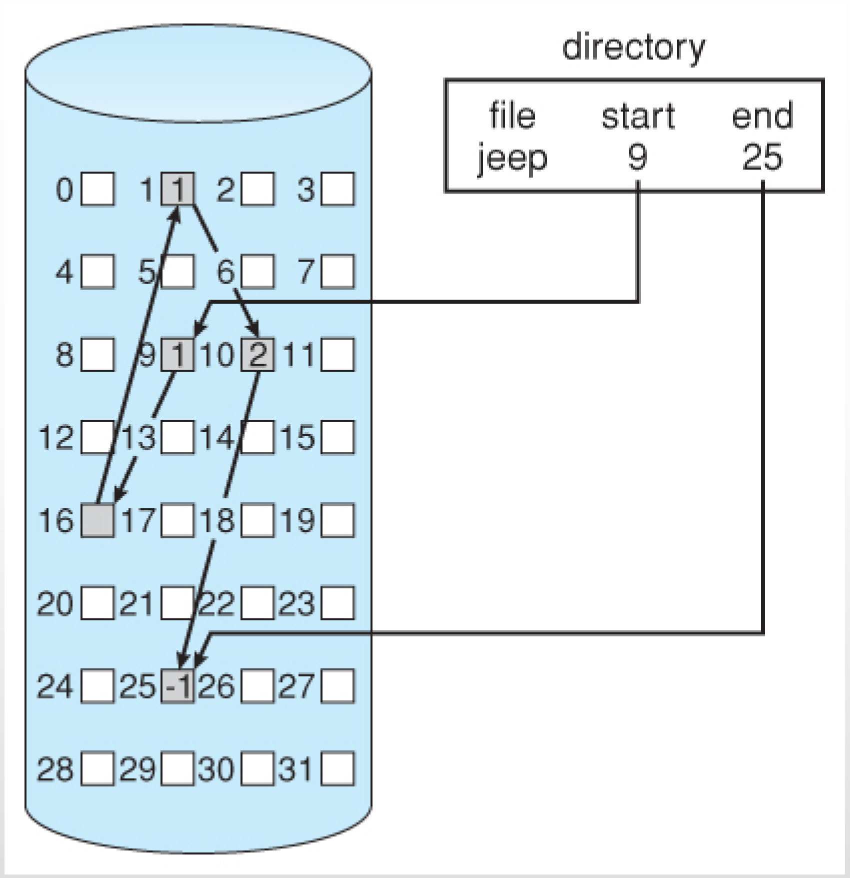
- *Image viewer* – Previous and next images are linked, hence can be accessed by next and previous button.
- *Previous and next page in web browser* – We can access previous and next url searched in web browser by pressing back and next button since, they are linked as linked list.
- *Music Player* – Songs in music player are linked to previous and next song. you can play songs either from starting or ending of the list.

Applications of linked list in computer science –

- Implementation of stacks and queues
- Implementation of graphs : Adjacency list representation of graphs is most popular which is uses linked list to store adjacent vertices.
- Dynamic memory allocation : We use linked list of free blocks.
- Maintaining directory of names

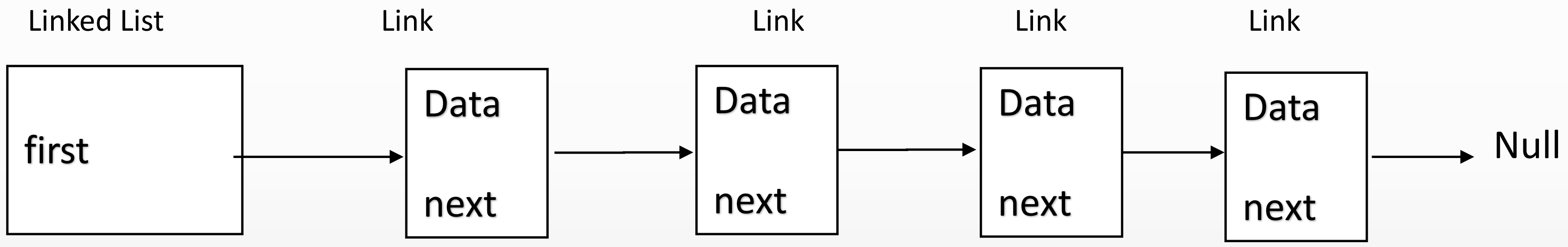
Linked Allocation in File System





Linked List

Linked lists are probably the second most commonly used general purpose storage structures after arrays.



- In a linked list each data item is embedded in a link.
- There are many similar links.
- Each link object contains a reference to the next link in the list.
- In a typical application there would be many more data items in a link.

Operations

- Mainly the following operations can be performed on a linked list.

  - Find

    Find a link with a specified key value.

  - Insert

    Insert links anywhere in the list.

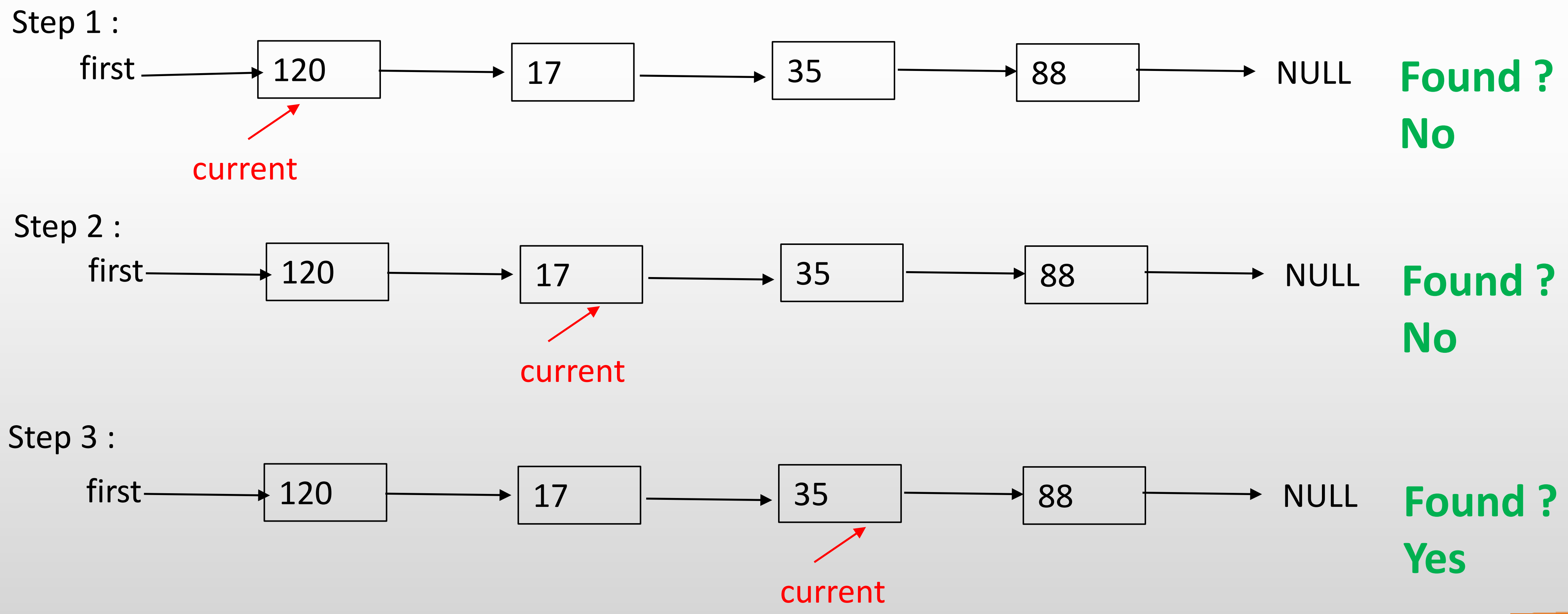
  - Delete

    Delete a link with the specified value.

Operations - Find

Start with the first item, go to the second link, then the third, until you find what you are looking for.

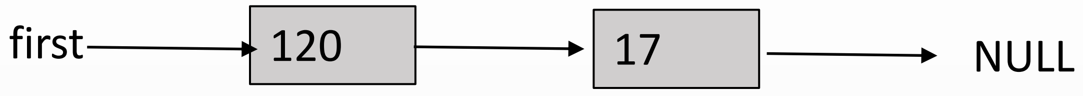
Ex: **Find Item 35**



Operations – Insert

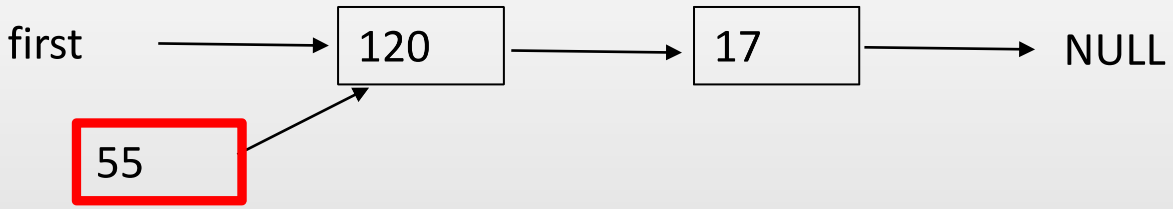
Inserting an item at the beginning of the list

Before inserting

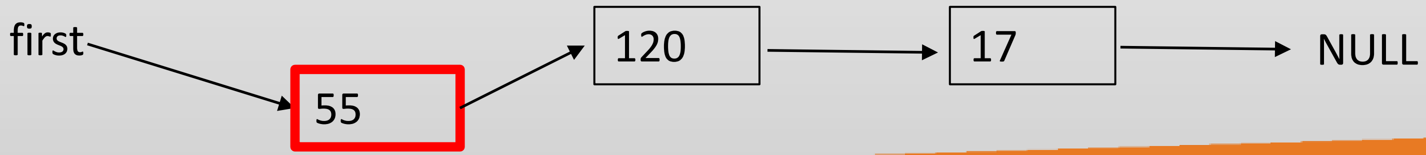


Step 1 : create a new link

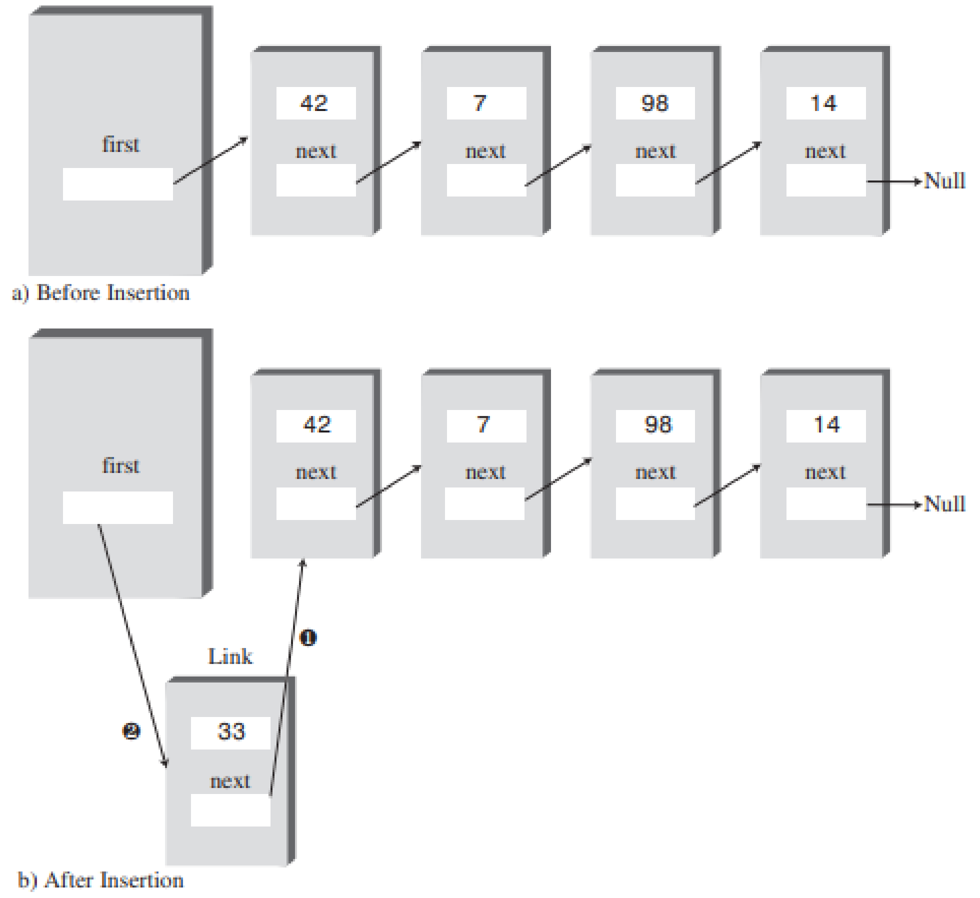
Step 2 : ‘next’ field of the new link points to the old first link



Step 3 : ‘first’ points to the newly created link



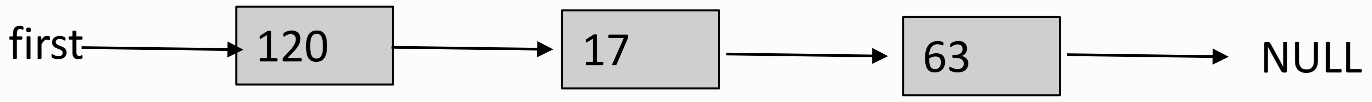
InsertFirst()



Operations - Insert

Inserting an item in the middle of the list

Before inserting



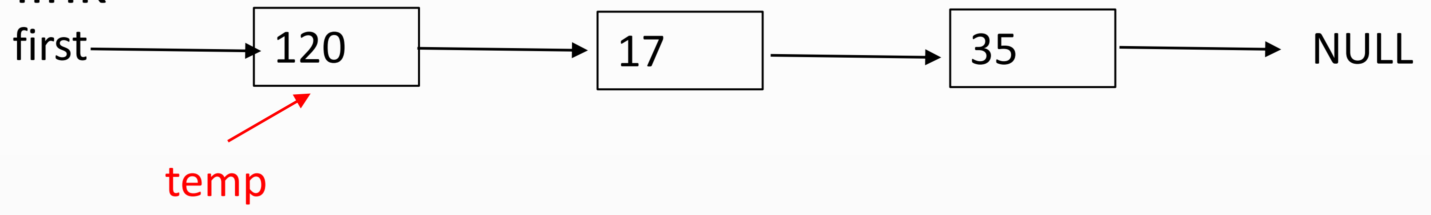
Question:

What steps need to be followed if a new link is inserted after the link ‘17’ ?

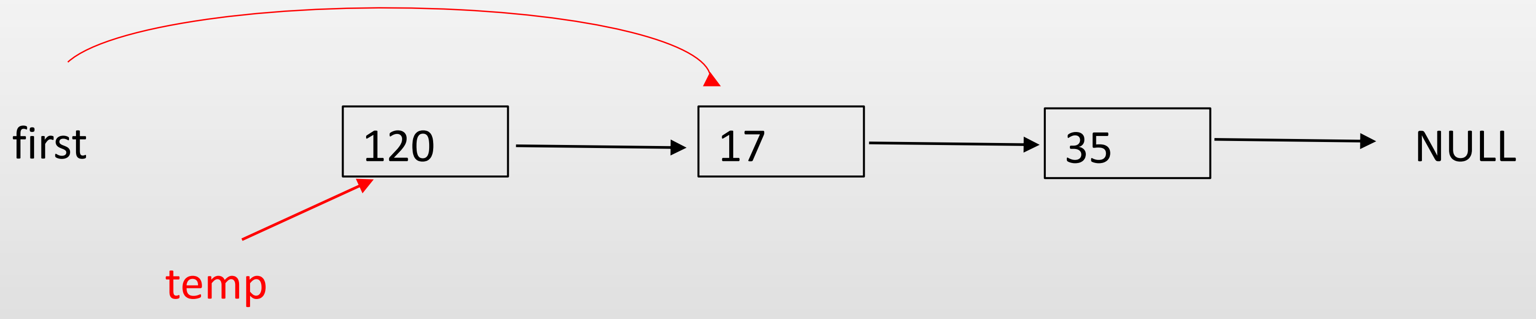
Operations - Delete

Deleting an item from the beginning of the list

Step 1 : Save reference to first link

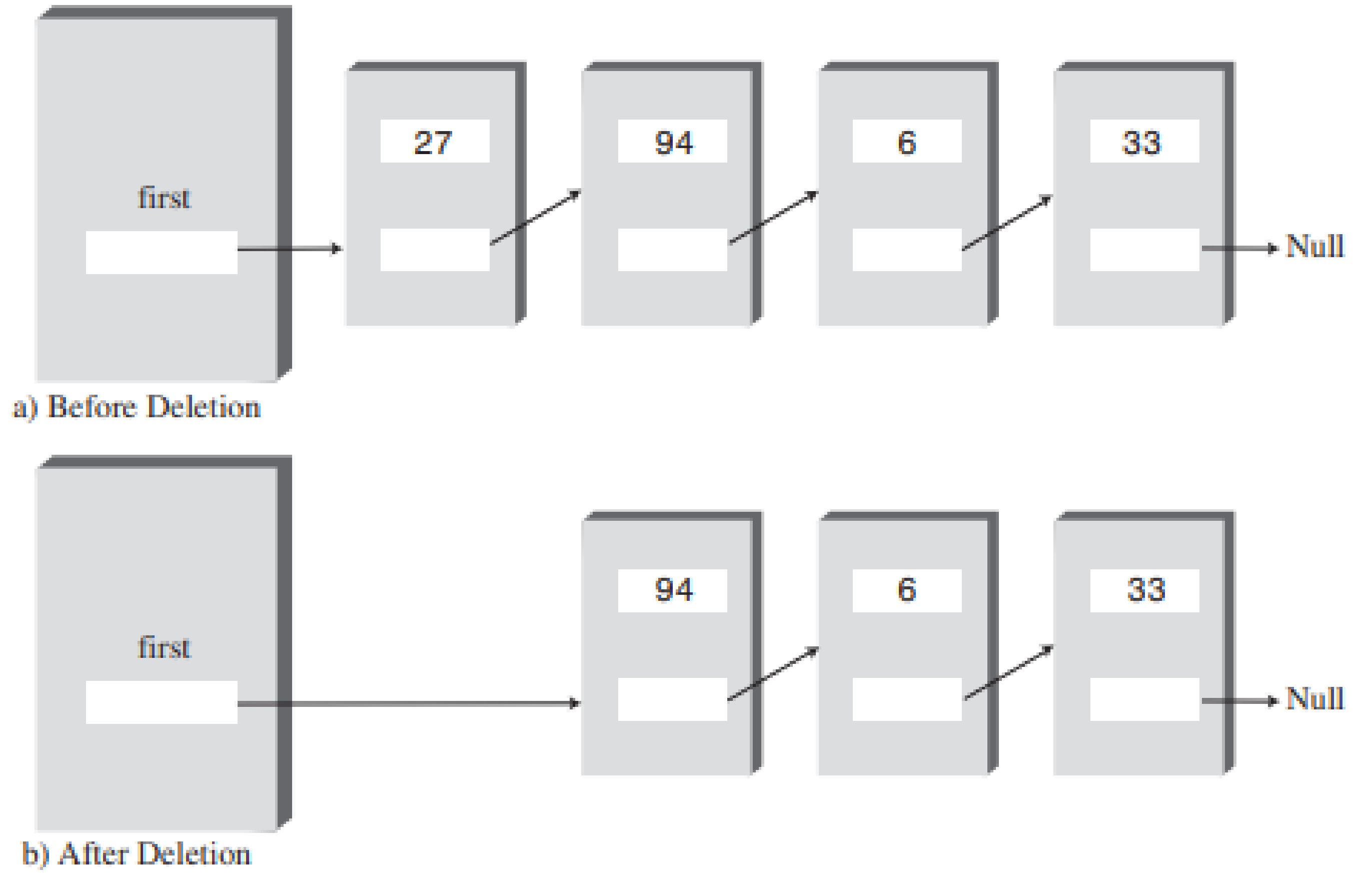


Step 2 : Disconnect the first link by rerouting first to point to the second link



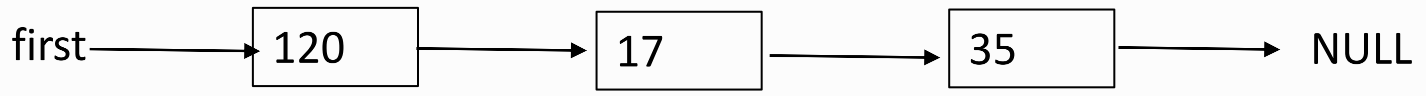
Step 3 : Return the deleted link (temp)

deleteFirst()



Operations - Delete

Deleting a given item from the list



Question:

What steps need to be followed to delete the link ‘17’ ?

Linked List - Implementation

Link Class

- In a linked list, a link is an object of a class called something like “**Link**”.

- There are many similar links in a linked list.

- Each link contains Data Items and a reference to the next link in the list.

```
class Link {
      public int iData;    // data item
      public Link next;    // reference to the next link

      public Link(int id) { // constructor

                iData = id;
                next  = null;
       }
       public void displayLink() {  // display data item

                System.out.println(iData);
       }
}
```

Linked List - Implementation

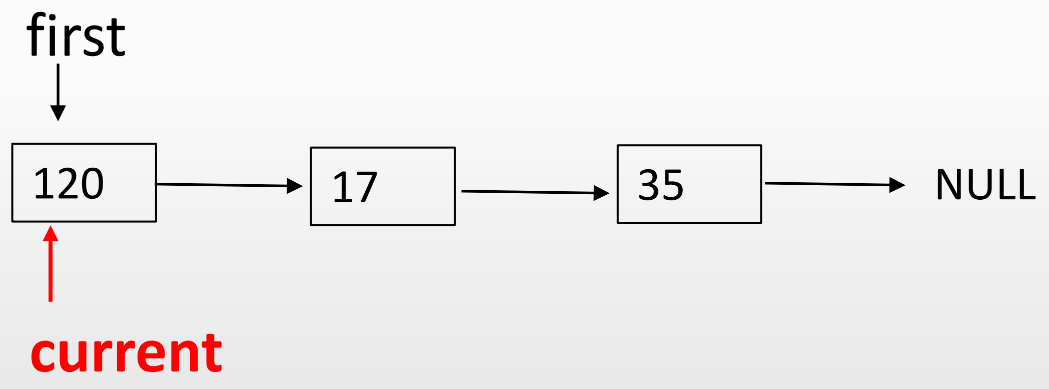
Link List Class

- The LinkList class contains only one data item, a reference to the first link on the list called **‘first’**.
- It is possible to find the other links by following the chain of references from ‘**first**‘, using each link’s next field.

```
class LinkList {
    private Link first;

     public LinkList() {     //constructor

           first = null;
     }
     public boolean isEmpty() {  // true if list is empty

           return (first == null);
     }
     // ……………………        other methods
}
```

Linked List - Implementation

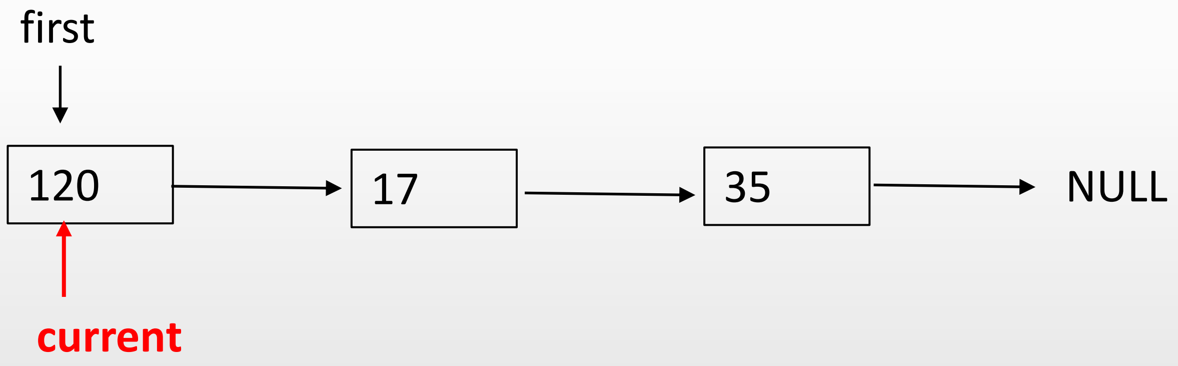
Link List Class – Contd.



```
class LinkList {
     private Link first;
    public  LinkList() { //constructor
                first = null;
    }
    public boolean isEmpty() {  // true if list is empty
                return (first == null);
    }
     public void displayList()     {
               Link current = first;
               while (current != null) {
                     current.displayLink();
                     current = current.next;
               }
               System.out.println(" ");
     }
}
```

Linked List - Implementation

Link List Class – Contd.



```
class LinkList {
    private Link first;
    public LinkList()  {  //constructor
            first == null;
    }
    public boolean isEmpty()  { // true if list is empty
            return (first == null);
    }
     public void displayList() {
            Link current = first;
            while (current != null) {
                    current.displayLink();
                    current = current.next;
            }
            System.out.println(" ");
    }
}
```

Linked List - Implementation

Link List Class – Contd.

```
class LinkList {
      private Link first;
      public LinkList() {  //constructor

                first = null;
      }
      public boolean isEmpty() { // true if list is empty

                return (first == null);
      }
      public void displayList() {

            Link current = first;
            while (current != null) {

                  current.displayLink();
                  current = current.next;
            }
            System.out.println(" ");
     }
```

```
// insertFirst Method
public void insertFirst(int id) {
            Link newLink = new Link(id);
            newLink.next = first;
            first = newLink;
  }
```

```
// deleteFirst Method
public Link deleteFirst()  {
            Link temp = first;
            first = first.next;
            return temp;
  }
}
```

Question 1

Write a program to

i) Create a new linked list and insert four new links.
ii) Display the list.
iii) Remove the items one by one until the list is empty.

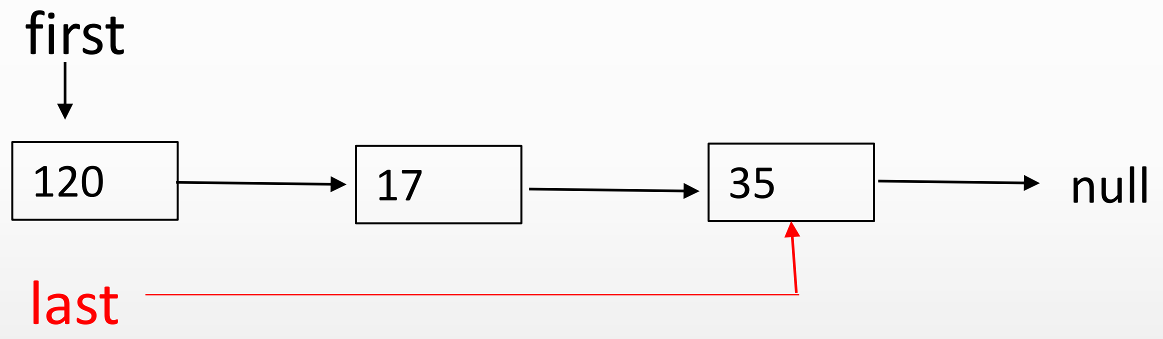
(Use the LinkList class created)

Answer1

```
class  myList {
        public static void main(String[] args)        {
                LinkList theList  = new LinkList();    // create a new list

                 theList.insertFirst(23);    // insert four items
                 theList.insertFirst(89);
                 theList.insertFirst(12);
                 theList.insertFirst(55);

                theList.displayList();          //display the list

                while( !theList.isEmpty() ) {   // delete item one by one

                                Link aLink = theList.deleteFirst();
                                System.out.print("Deleted ");
                                aLink.displayLink();
                }
        }
}
```

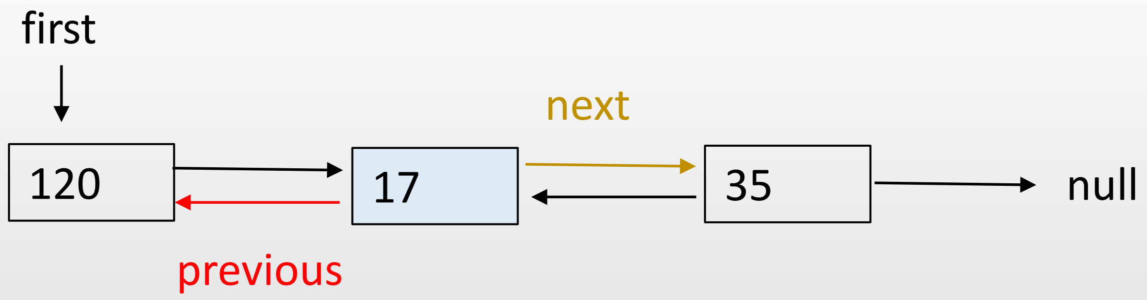
Double-Ended List

A double-ended list is similar to an ordinary linked list with an additional reference to the last link.



Doubly Linked List

A doubly linked list allows to traverse backwards as well as forward through the list.
Each link has two references.



References

Mitchell Waite, Robert Lafore, Data Structures and Algorithms in Java, 2nd Edition, Waite Group Press,1998.