Linked List Class

```
class LinkedList {
   private:
      Node* head;
   public:
      LinkedList(); /* Constructor */
      ~LinkedList(); /* Destructor */

      void traverse();
      Node* search(int val);
      void insertNode(int leftValue, int value);
      void deleteNode(int value);
};
```

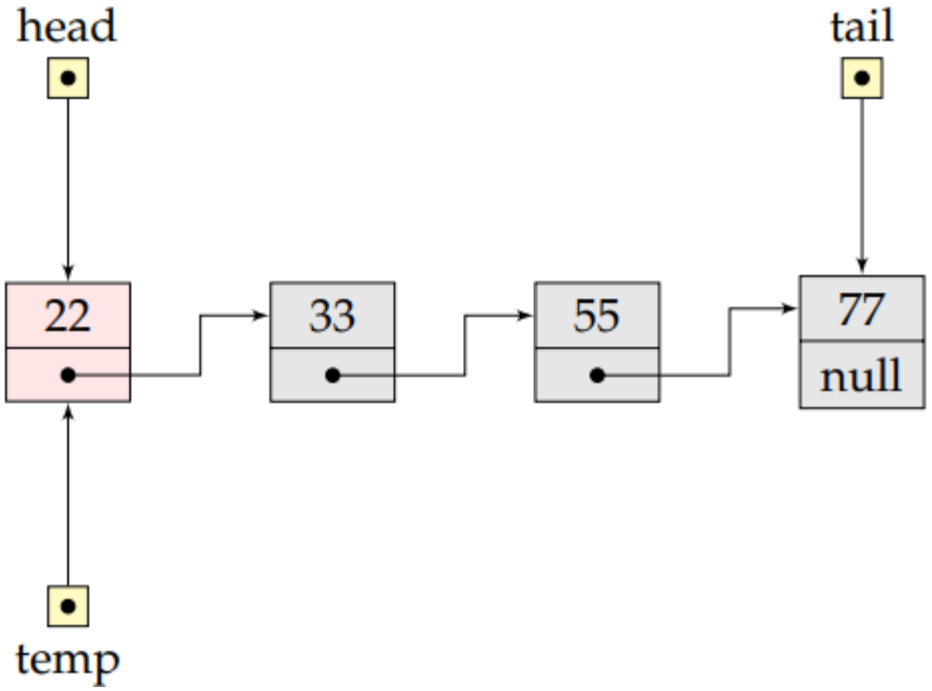
Linked List: Deleting a Node

- First, let's see if the node we want to delete even exists in the list.

```
Node *temp=search(value);
if (temp == 0)
    return;
else "remove Node" // i.e. deleteNode(value);
```

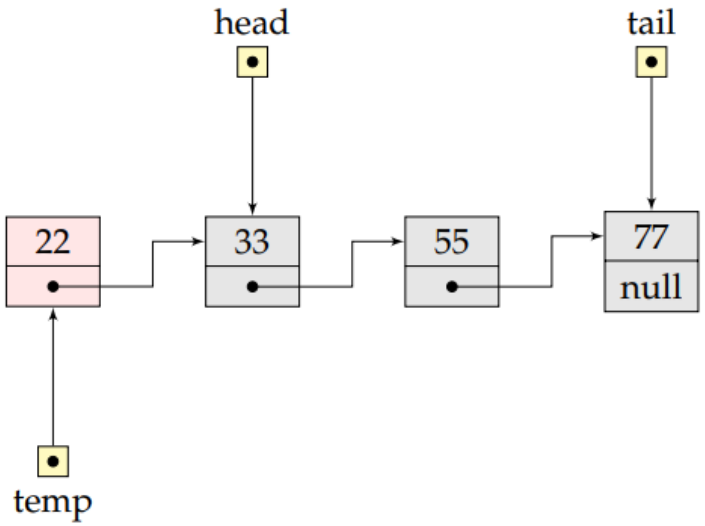
Linked List: Deleting a Node

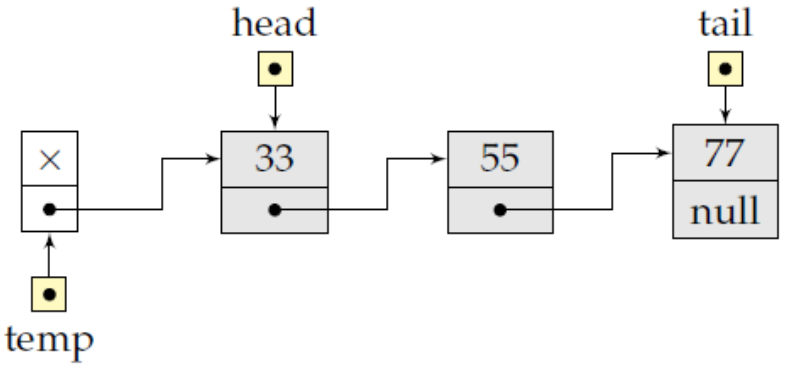
```
if (head->data == value)
{
    Node* temp = head;
    head = head->next;
    delete temp;
}
```



Linked List: Deleting a Node

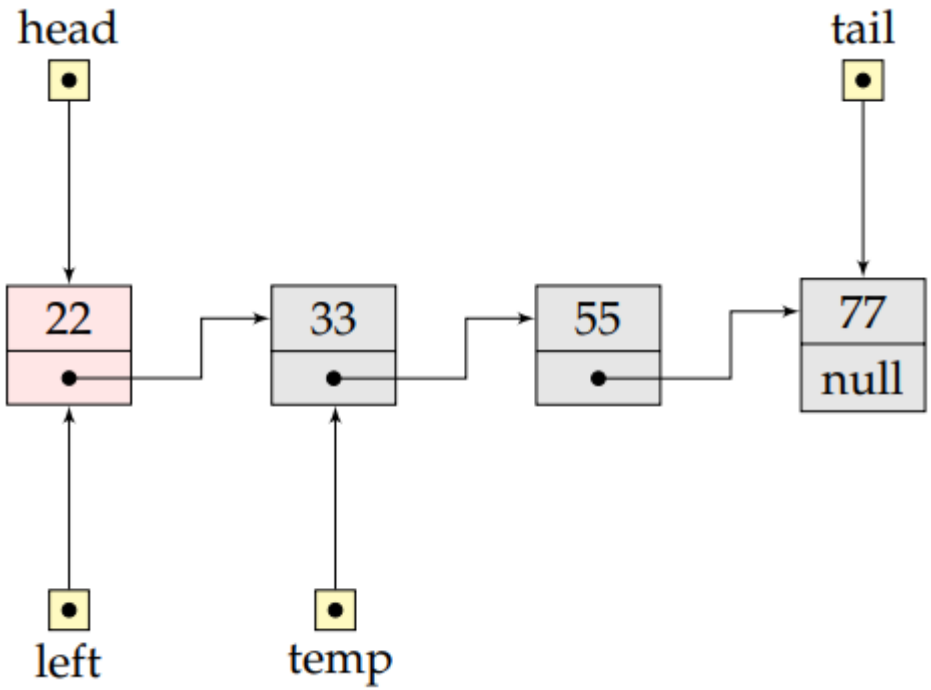
```
if (head->data == value)
{
    Node* temp = head;
    head = head->next;
    delete temp;
}
```





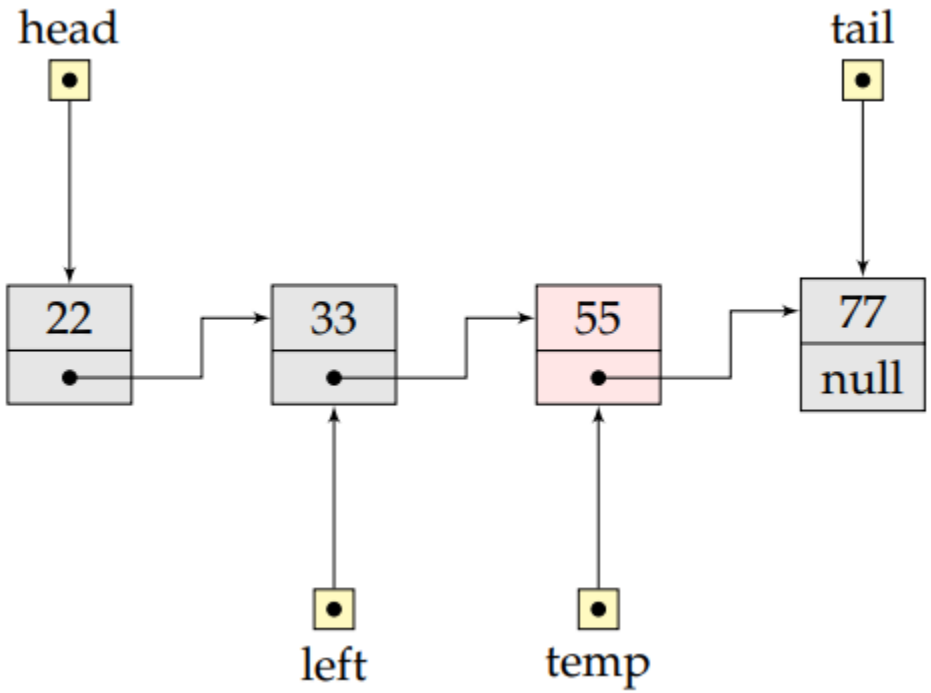
Linked List: Deleting a Node

```
else /*either tail node or middle node */
{
   Node* left = head;
   Node* temp = left->next;
   bool isFound = false;

   while (temp != 0 && isFound != true)
   {
      if (temp->data == value)
      {
          if (temp->next == 0)  /* tail node */
          {
             left->next = 0;
             tail = left;
          }
          else
          {
             left->next = temp->next;
          }
          delete temp;
          isFound = true;
      }
      else
      {
         left = temp;
         temp = temp-> next;
      }
   }
}
```



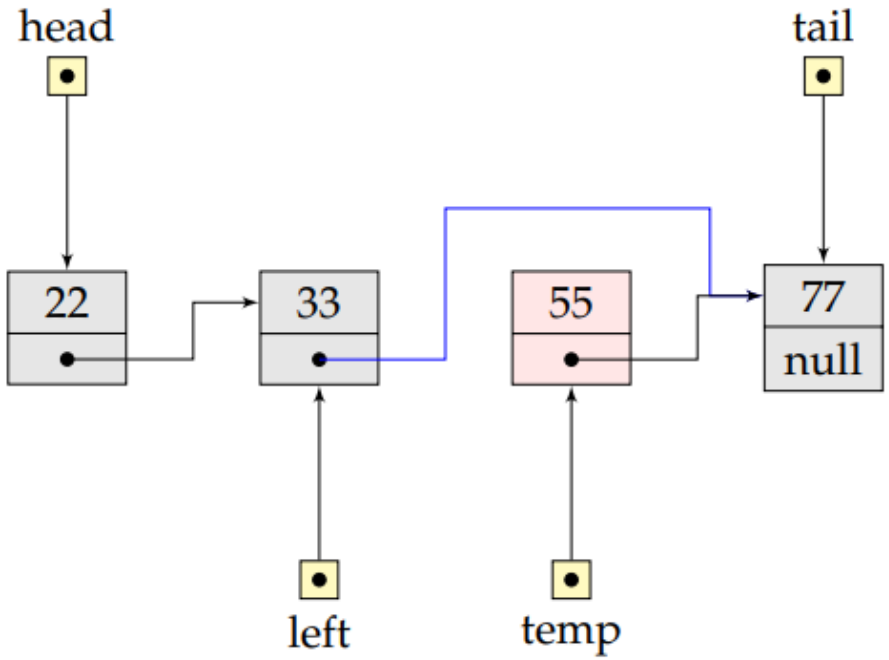
Linked List: Deleting a Node

```
else /*either tail node or middle node */
{
   Node* left = head;
   Node* temp = left->next;
   bool isFound = false;

   while (temp != 0 && isFound != true)
   {
      if (temp->data == value)
      {
          if (temp->next == 0)  /* tail node */
          {
             left->next = 0;
             tail = left;
          }
          else
          {
             left->next = temp->next;
          }
          delete temp;
          isFound = true;
      }
      else
      {
         left = temp;
         temp = temp-> next;
      }
```



Linked List: Deleting a Node

```
else /*either tail node or middle node */
{
   Node* left = head;
   Node* temp = left->next;
   bool isFound = false;

   while (temp != 0 && isFound != true)
   {
      if (temp->data == value)
      {
          if (temp->next == 0)  /* tail node */
          {
             left->next = 0;
             tail = left;
          }
          else
          {
             left->next = temp->next;
          }
          delete temp;
          isFound = true;
      }
      else
      {
         left = temp;
         temp = temp-> next;
      }
   }
}
```



Linked List: Deleting a Node

```
void LinkedList::deleteNode(int value) {
   if (head->data == value) {
      Node* temp = head;
      head = head->next;
      delete temp;
    }
    else { /*either tail node or middle node */
       Node* left = head;
       Node* temp = left->next;
       bool isFound = false;
       while (temp != 0 && isFound != true) {
          if (temp->data == value) {
             if (temp->next == 0) { /* tail node */
                left->next = 0;
                tail = left;
             }
             else {
                left->next = temp->next;
             }
             delete temp;
             isFound = true;
          }
          else {
             left = temp;
             temp = temp->next;
}}}}
```

Linked List: Destroying a List

- Use the destructor
- Since each node is created dynamically, each one must be deleted
- Requires traversal – deleting each node one at a time
- Use temporary pointer to assist in deletion
- Use the head pointer to traverse
- Head should point to null when finished

Linked List: Destroying a List

```
~LinkedList()
{
    Node *next;
    while(head != NULL)
    {
        next = head->next;
        delete head;
        head = next;
    }
}
```

Linked List: Destroying a List

```
~LinkedList()
{
   Node *current = head;
   while( current != 0 ) {
      Node *next = current->next;
      delete current;
      current = next;
   }
   head = 0;
}
```

Questions