delete

Wednesday, September 6, 2017 10:29 AM

Deleting

Watch for the following when deleting:

1) An attempt to remove a node from an empty list
2) Deleting the only node from a one node linked list. Both head and tail are set to null
3) Removing the first node requires updating the head.
4) Removing the last node requires updating of the tail.
5) Removing a node from the middle requires that the previous node's next member to contain the deleted node's next member's address.

When deleting, consider, deleting the first node, the last node and also any other node.

Example 11: ERROR -- Removing from a list

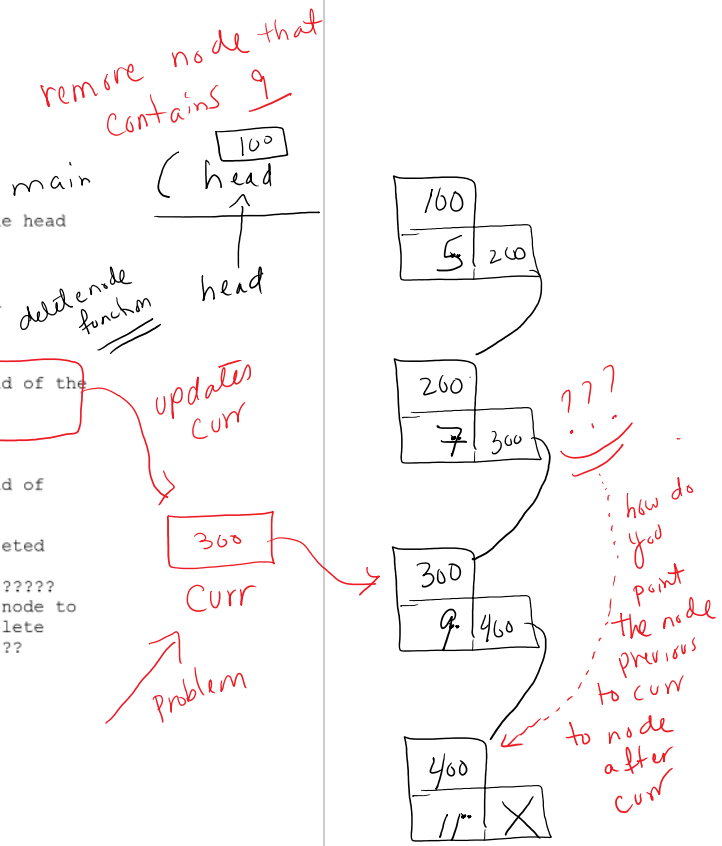
```
void deleteNode(int item, node *&head){
       node *temp=NULL, *curr=head;

       //Make sure that there is  list
       if (head!=NULL){
            //if the first item is a match
            if ( (head->a==item){
                temp=head;        //get ready to take out the head
                head=head->next;  //head is updated
            }else{
                curr=curr->next;
                //Start looking from the second item forward
                //Make sure that there is a second item
                //Move curr as long as it is not found
                while (curr!=NULL && curr->a!=item)
                    //Notice that we are always looking ahead of the
                    //link that we are in
                    curr=curr->next;

                //If item has been found then curr should
                //not be null because the found item is ahead of
                //where curr is
                if (curr!=NULL){
                    //curr of the place that needs to be deleted
                    temp=curr;
                    //??????????????????????????????????????????
                    //error: How do we connect the previous node to
                    //one beyond the node that we want to delete
                    //??? previous curr->next= curr->next; ???

                }
            }

            //Only if item was found should we delete
            if(temp)
                delete temp;
     }
}
```



Example 12: Correcting previous version -- Removing from a list

```
void deleteNode(int item, node *&head){
       node *temp=NULL, *curr=head;

       //Make sure that there is  list
       if (head!=NULL){
            //if the first item is a match
            if ( (head->a==item){
                temp=head;         //get ready to take out the head
                head=head->next;  //head is updated
            }else{
                //Start looking from the second item forward
                //Make sure that there is a second item
                //Move curr as long as it is not found
                while (curr->next!=NULL && curr->next->a!=item)
                    //Notice that we are always looking ahead of the
                    //link that we are in
                    curr=curr->next;

                //If item has been found then curr->next should
                //not be null because the found item is ahead of
                //where curr is
                if (curr->next!=NULL){
                    //curr of the place that needs to be deleted
                    temp=curr->next;

                    //curr is where we are pointing to
                    //curr->next contains address of node to be deleted
                    //curr->next needs to skip to curr->next->next
                    curr->next=curr->next->next;
                }
            }

            //Only if item was found should we delete
            if(temp)
                delete temp;
    }
}
```