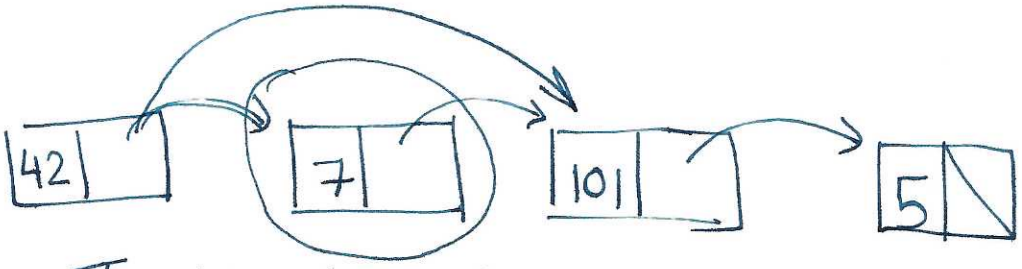
Deletions in linked lists

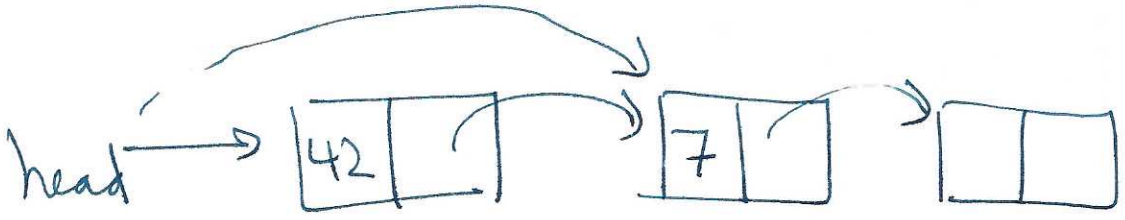


Is this laggy?

Deletion in linked lists

Delete 7.

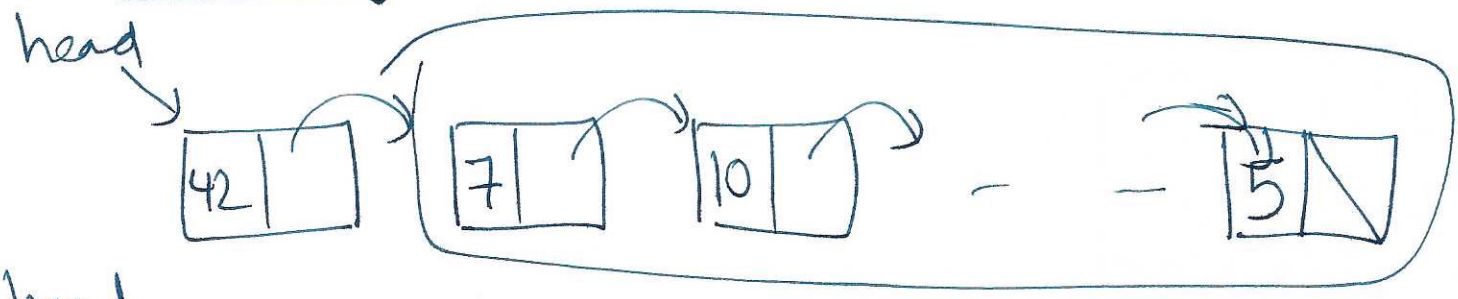
Need a pointer to PREVIOUS node.

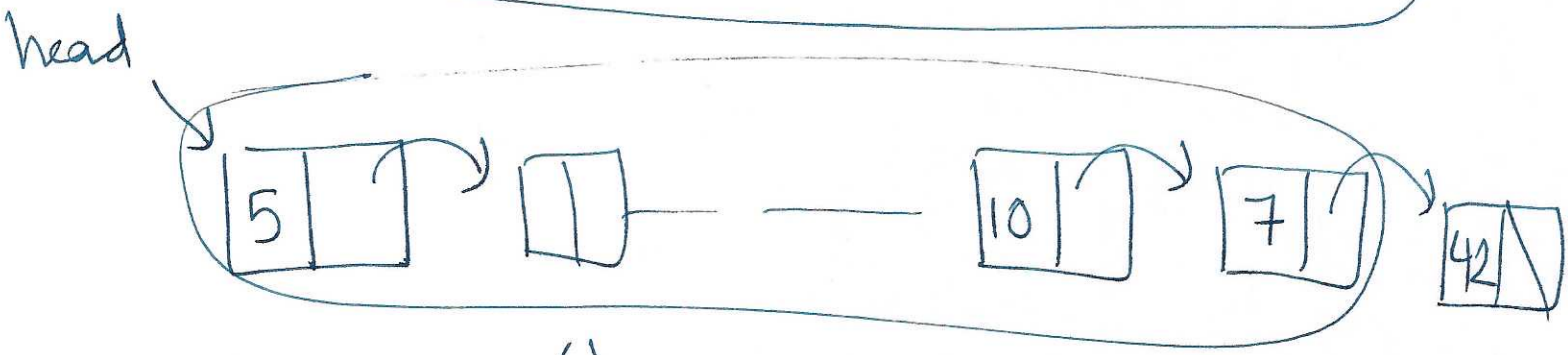


How to delete 42? head needs to be changed.

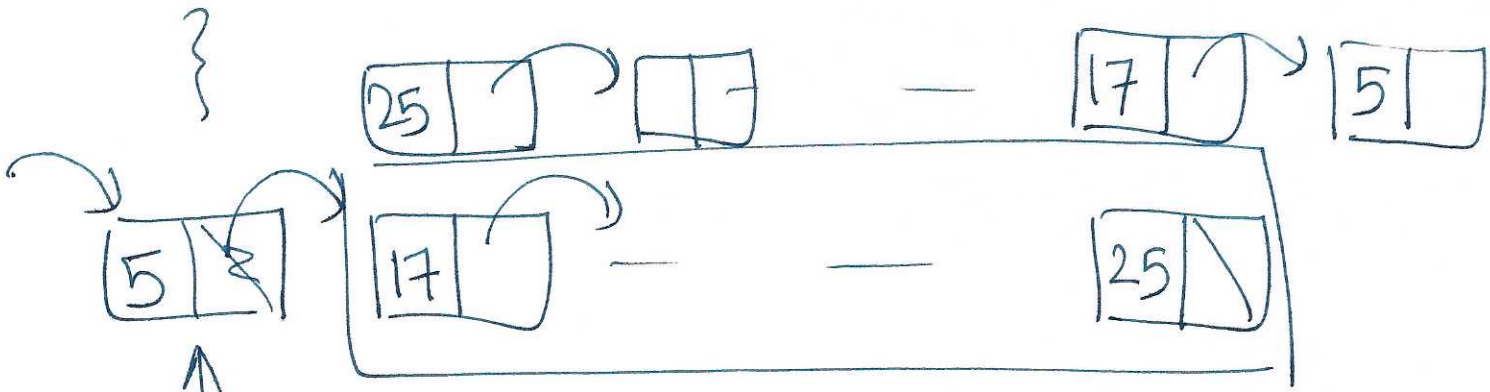
1) Dealing with the head
2) Needing pointer to PREV node to the node being deleted/processed

Reversing a linked list





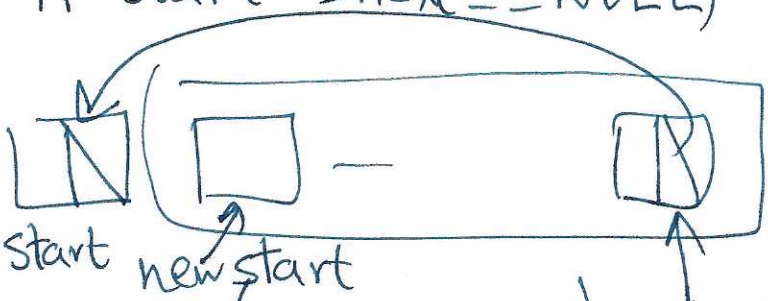

```
void reverse()
{

}
```



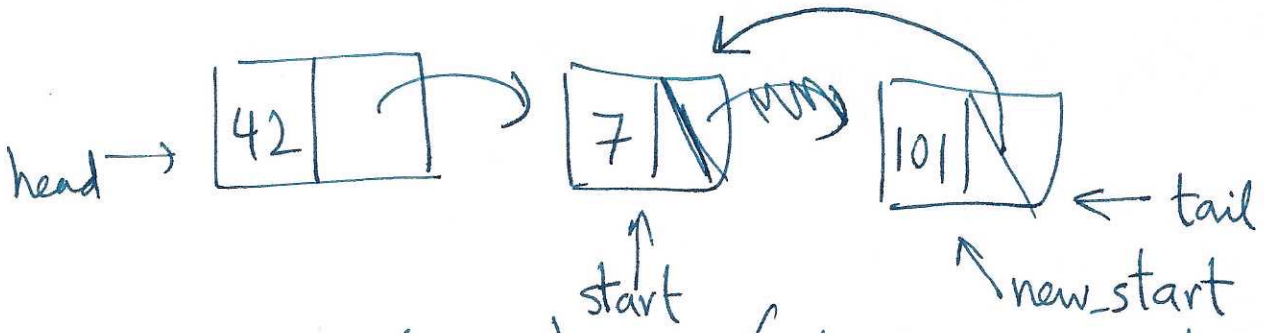
1) Reverse everything from start → next recursively

2) Get the ~~first~~ last node of the reversed part, ~~add~~ insert $ start at the END.

```
Node * reverse (Node* start)  (returns head of
{                                  reversed portion)
    if (start == NULL || start->next == NULL)
        return (start);
```




```
    ~~Node * new_head~~
    Node *new_start = reverse (start->next);
    Node *tail = getTail (~~start~~ new_start);
                    // get the last node from new_start
    tail -> next = start;
    start -> next = NULL;
    return new_start;
}

~~void reverse~~
void reverse()
{
head = reverse (head);
}
```

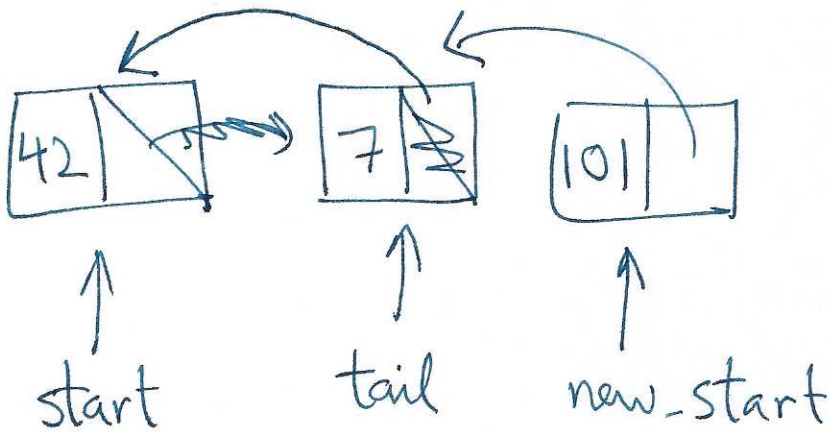


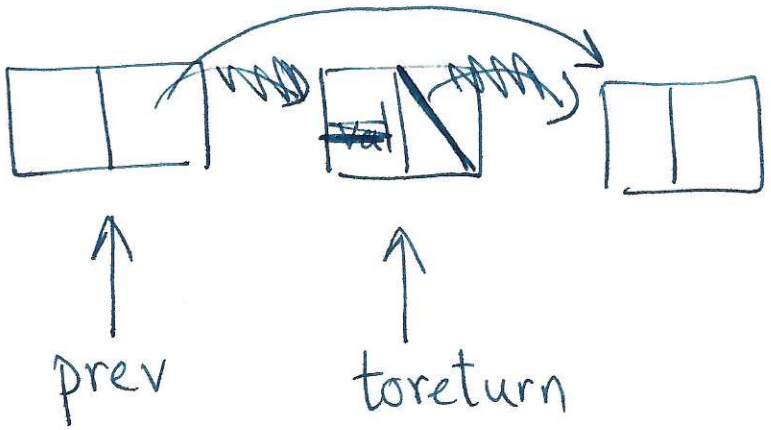
reverse(head) (start is head)

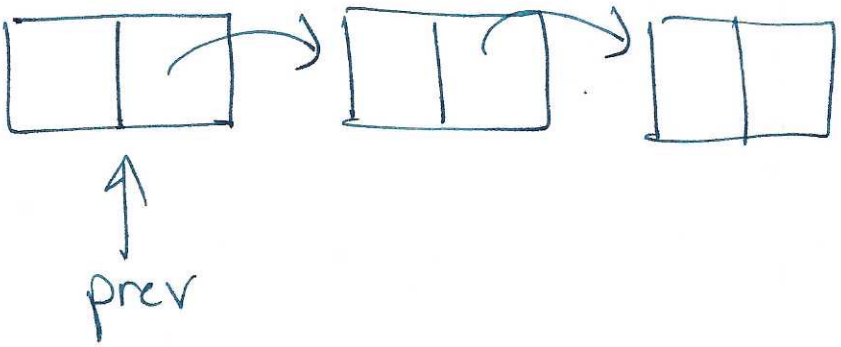
reverse(head->next) (start is head->next)

returns ptr to 101

reverse(head->next->next) (start is last node)







```
Node* delete(int val)
{
    // if head->data is val, delete head node and update head
    return delete(val, head);
}
```

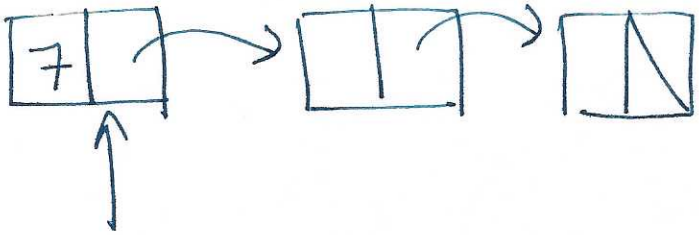
This deletes ONLY the FIRST occurrence of val.

Test: How to delete the LAST occurrence?

ALL

delete the k^{th} occurrence?

```
Node*delete(int val)
{
```



} Recursive delete!
deletes val from list starting from ~~delete(7)~~ prev->next

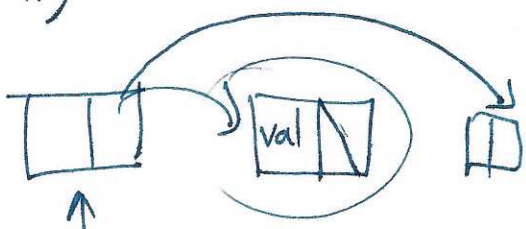
```
Node* delete(int val, Node* ~~start~~ prev)
{
    if (~~start~~ == NULL)        // list from ~~start~~ prev is empty
        return NULL;
```

~~if (start->val data == val)~~

```
    if (prev->next == NULL)
        return NULL;
```

prev->next
↓
data

```
    if (prev->next->data == val)   //now we delete
    {
        Node *toreturn
              = prev->next;
        prev->next = prev->next
                       ->next;
        toreturn->next = NULL;
        return toreturn;
    } else return delete(val, prev->next); }
```



Iteration