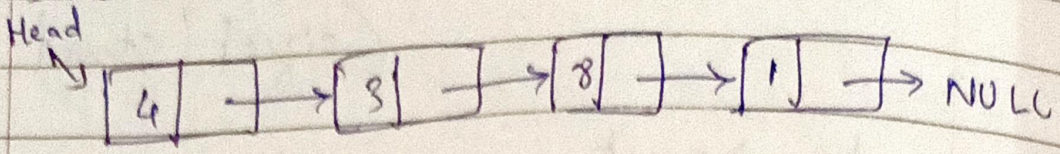
Deletion in a linked list



Cases :-

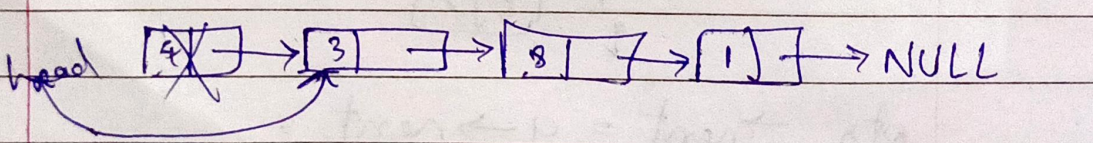
1. Deleting the first node
2. Deleting a node in between
3. Deleting the last node.
4. Delete a node with a given value.

⇒ After deleting any node, by any case, we would just need to free that extra node left after we disconnect it from the list.

⇒ As we free a node, it remove its reserved location in the heap.

I. Deleting the first Node

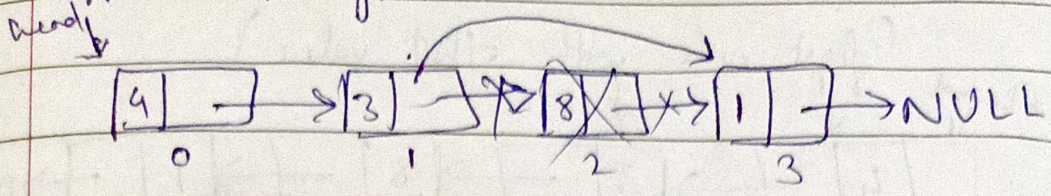
```
struct Node * ptr = head ;
head = head -> next
free(ptr) ;
```



Time Complexity = O(1)

Case 2:- Deleting a node in between

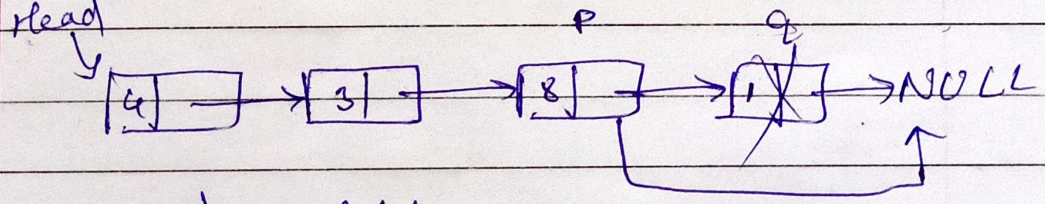
inden is given.




```
inden = 2;
struct Node * p = head;
while ( p -> q != (inden - 1))
{
        p = p -> next ; i++ ;
}
struct Node * q = p -> next;
  p -> next = q -> next;
    free (q)
```

Time Complexity :- O(n)

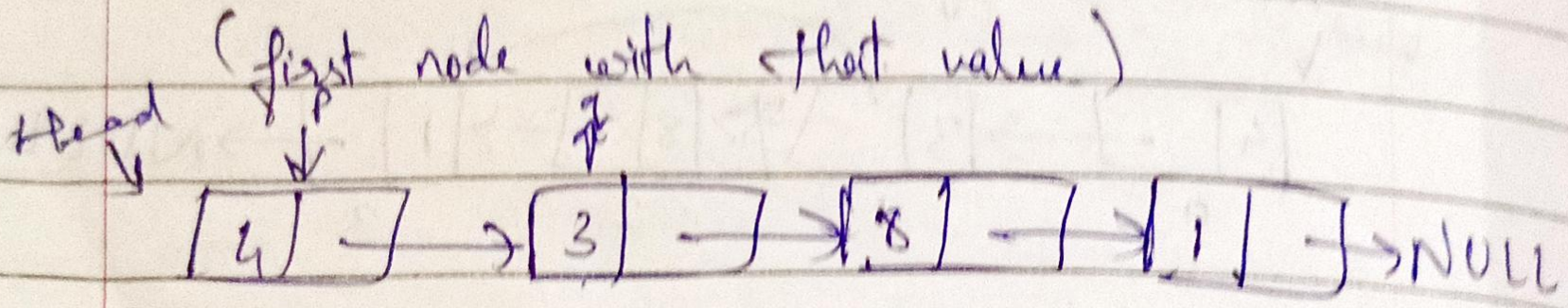
Case 3:- Delete last node :-




```
p -> next = NULL
free (q);
```

Time Complexity :- O(n)

Case 6:- Delete a node with given value :-



value = 8

```
while (q -> data != value) {
    p = p -> next ;
    q = q -> next ;
}
p -> next = q -> next ;
free (q) ;
```

Time Complexity = O(n)