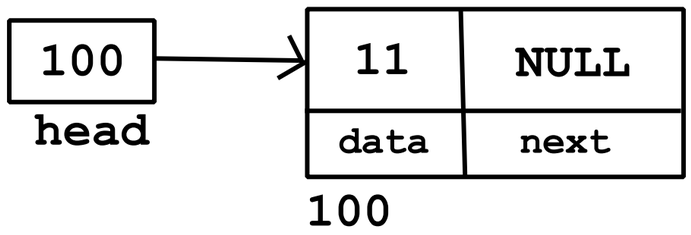
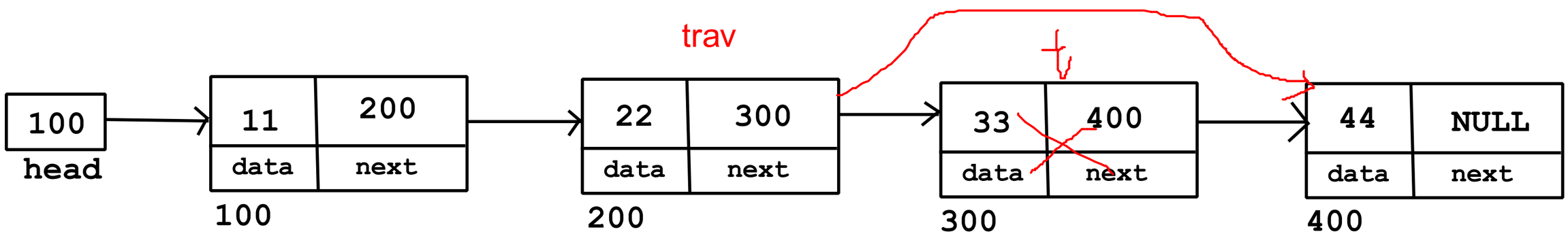
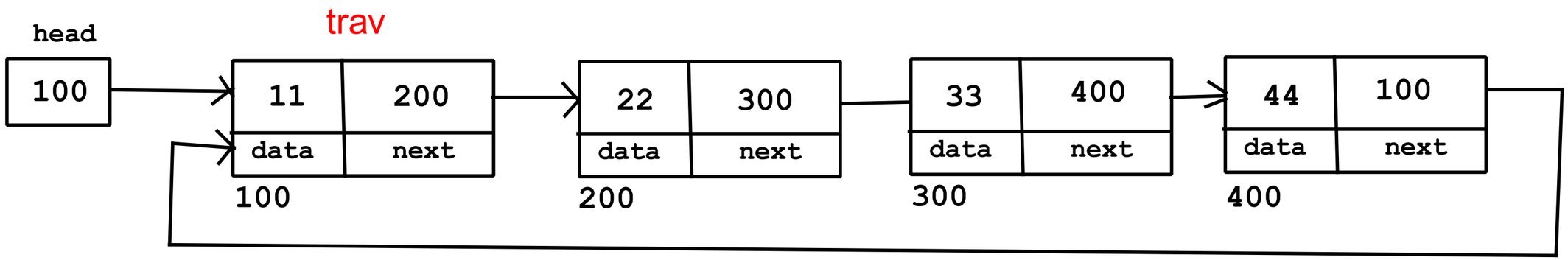
Linked List - Delete Node at Position




```
//1. if list is empty
        //print msg
//2. if list has only one node
        // delete head node
        // add assign NULL to head
//3. if list has multiple nodes
        //a. traverse till pos - 1 node
        //b. take backup of pos node
        //c. create link between pos -1 node and pos + 1 node
        //d. delete backuped node                                  pos = 3
```



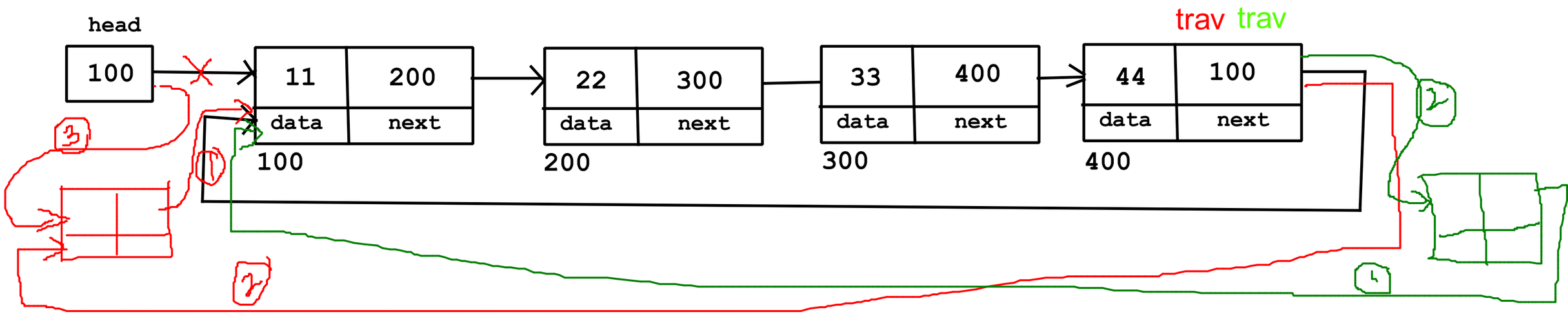
Singly Circular Linked List - Traverse



1. if list is empty
   print msg
2. if list is not empty
   a. start at head
   b. print / visit current node
   c. go on next node
   d. repeat step b and c till last node

```
void display(void)
{
        if(is_empty())
                printf("LIst is empty\n");
        else
        {
                node_t *trav = head;
                do{
                        printf("%d", trav->data);
                        trav = trav->next;
                }while(trav != head);
        }
}
```

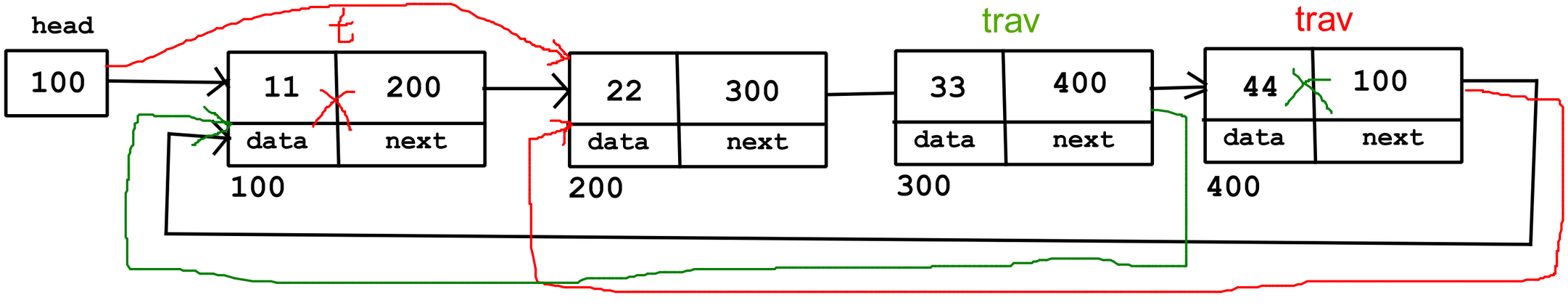
Singly Circular Linked List - Add Node



Add first:
0. create node
1. newnode->next = head
2. traverse till last node
3. add newnode into next of last node
4. add newnode into head

Add last:
0. create node
1. traverse till last node
2. add newnode into next of last node
3. add head into next of newnode

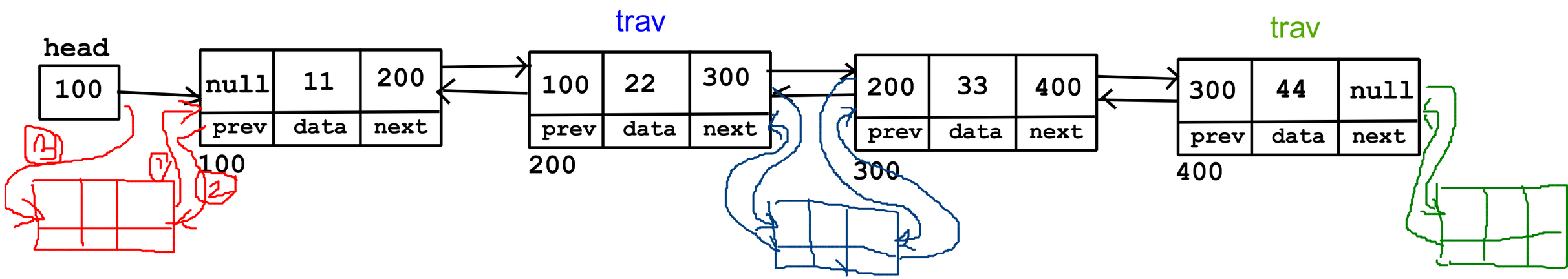
Singly Circular Linked List - Delete Node



Delete first:
1. take backup of first node
2. move head to second node
3. traverse till last node
4. add head into last node next
5. delete temp

Delete last:
1. Traverse till second last node
2. delete last node through trav->next
3. add head into last node(trav)

Doubly Linear Linked List



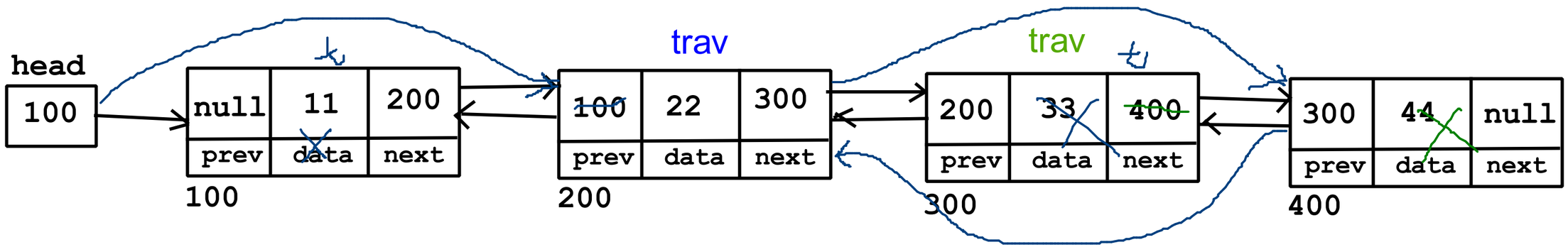
Display:
1. start at head
2. print data of current node
3. go on next node
4. repeat step 2 and 3 till last node

Add first:
1. create node
2. newnode->next = head
3. head->prev = newnode
4. head = newnode

Add last:
1. create node
2. traverse till last node
3. trav->next = newnode
4. newnode->prev = trav

Add position:
1. create node
2. traverse till pos -1 node
3. newnode->prev = trav
4. newnode->next = trav->next
5. trav->next->prev = newnode
6. trav->next = newnode

Doubly Linear Linked List

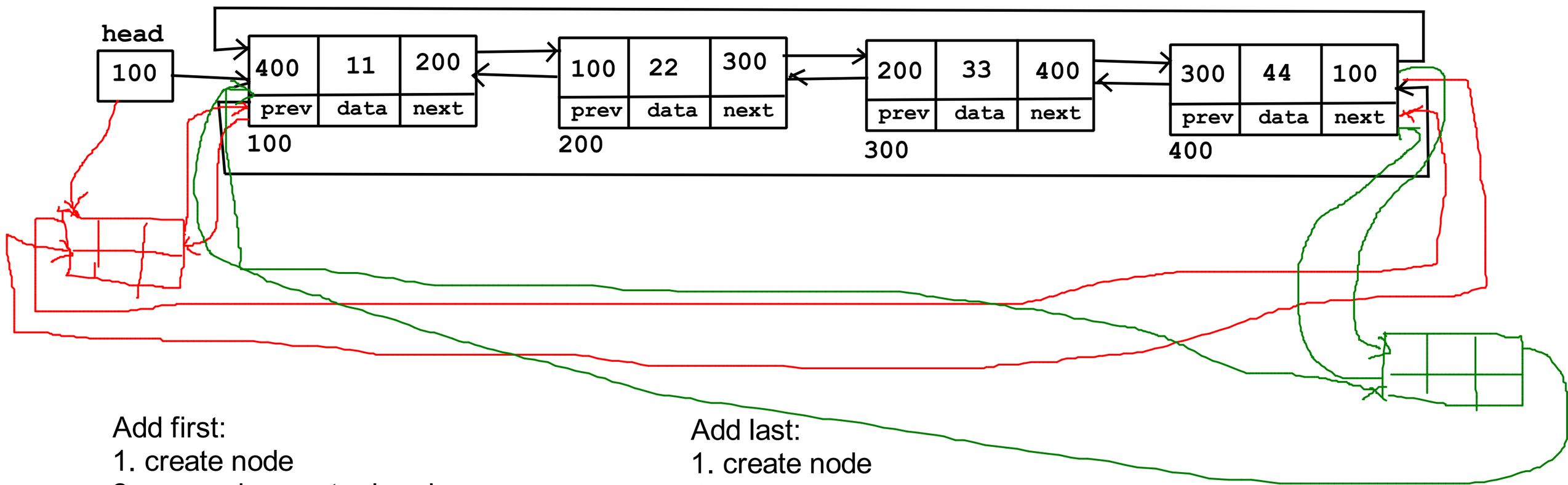


Delete first:
1. take backup of first node
2. head = temp->next
3. head->prev = NULL
4. free(temp)

Delete position:
1. traverse till pos - 1 node
2. take backup of pos node
3. trav->next = temp->next
4. temp->next->prev = trav
5. free(temp)

Delete last:
1. traverse till second last node
2. free(trav->next)
3. trav->next = NULL

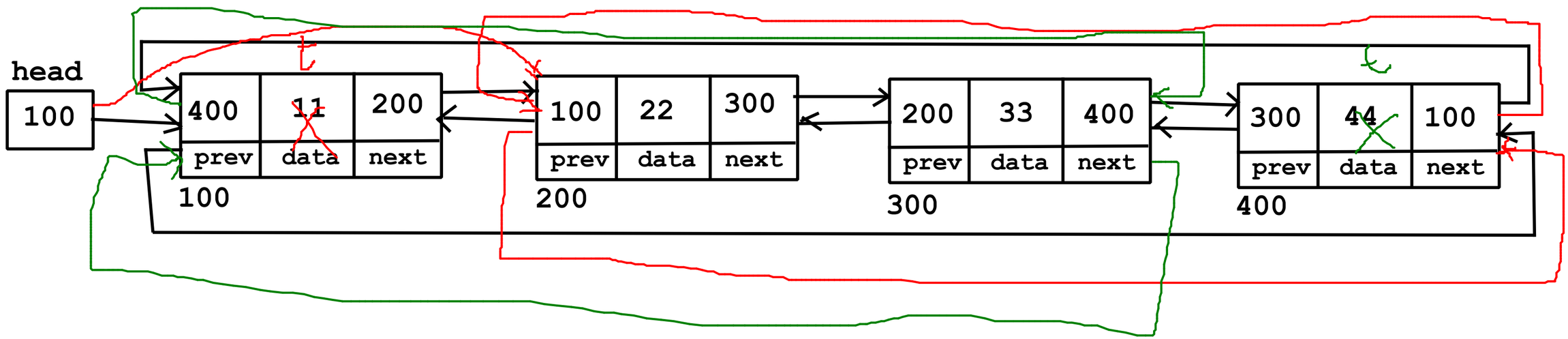
Doubly Circular Linked List



Add first:
1. create node
2. nenwode->next = head
3. newnode->prev = head->prev
4. head->prev->next = newnode
5. head->prev = newnode
6. head = newnode

Add last:
1. create node
2. newnode->next = head
3. newnode->prev = head->prev
4. head->prev->next = newnode
5. head->prev = newnode

Doubly Circular Linked List



Delete first:
1. take backup of first node
2. head = temp->next
3. head->prev= temp->prev
4. temp->prev->next = head
5. free(temp)

Delete last:
1. Take backup of last node
2. temp->prev->next = head
3. head->prev = temp->prev
4. free(temp)