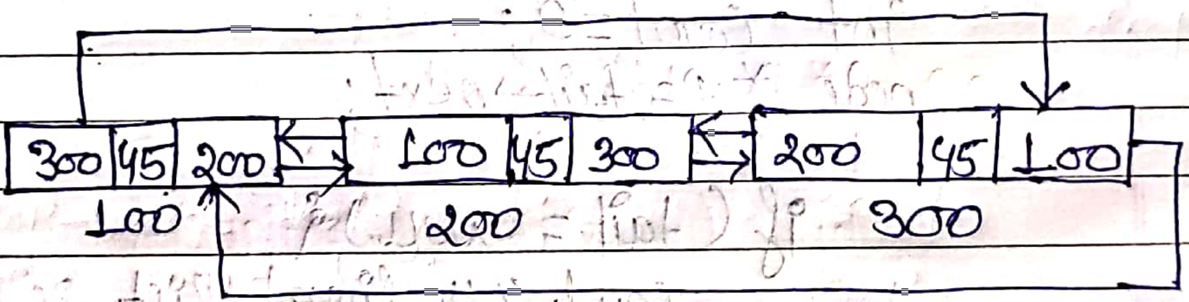
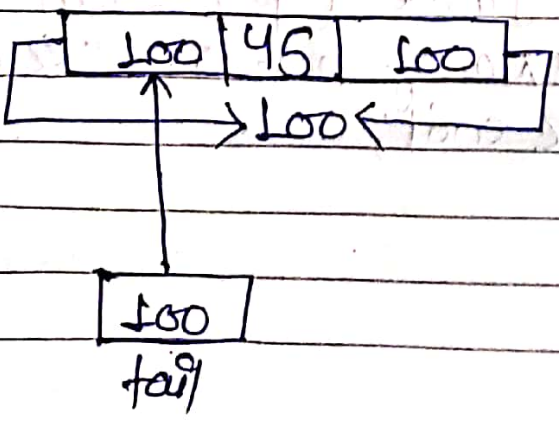
Circular Doubly linked list

Circular doubly linked list is similar to the doubly linked list except that the last node of the circular doubly linked list pointers points to the first node and the first node of the circular doubly linked list points to the last node.

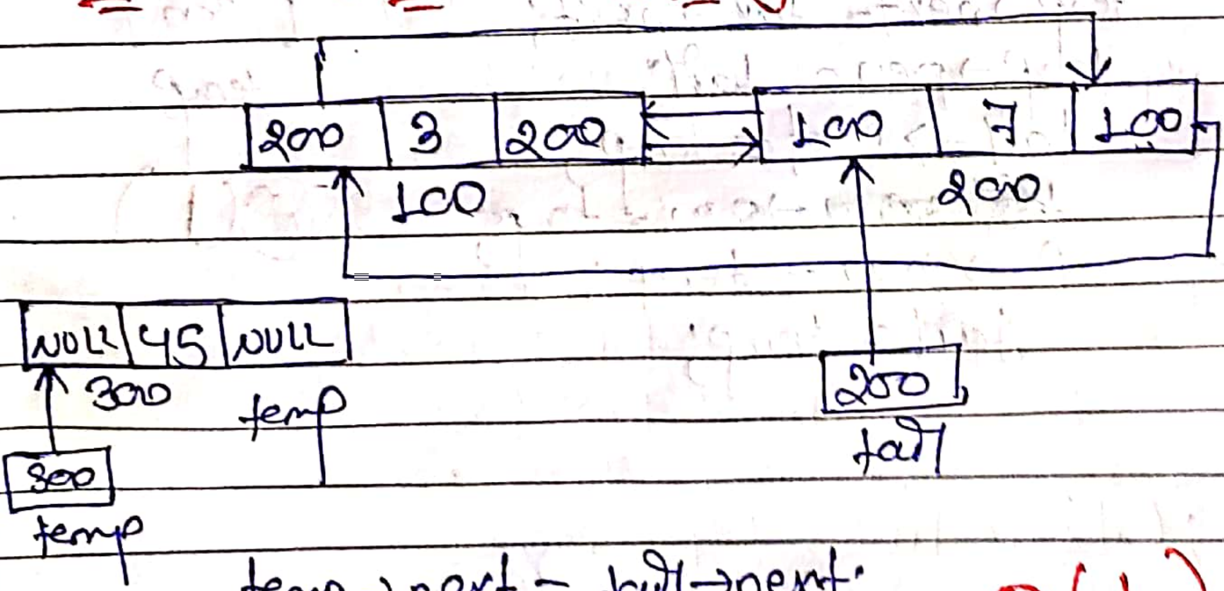


```
struct node {
    struct node * prev;
      int data;
    struct node * next;
    };
```



```
struct node *circularDoubly(node *tail, int data){
    node *temp = new node();
        temp -> data = data;
        temp -> next = temp;
        temp -> prev = temp;
            tail = temp;
        return tail;
    }
}
```

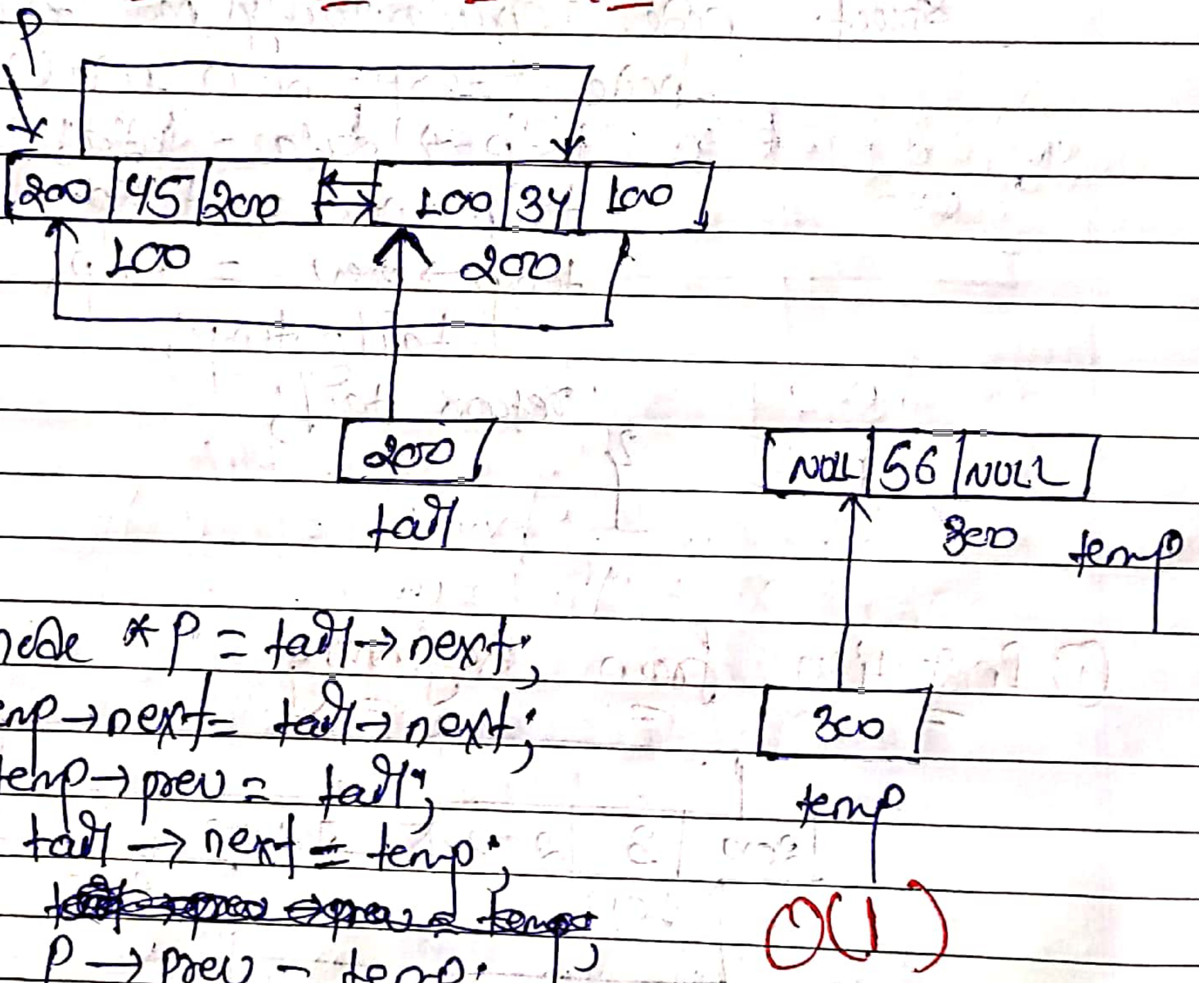
① Insertion from Beginning



```
temp -> next = tail -> next;
temp -> prev = tail;
tail -> next = temp;
temp -> next -> prev = temp;
```

O(1)

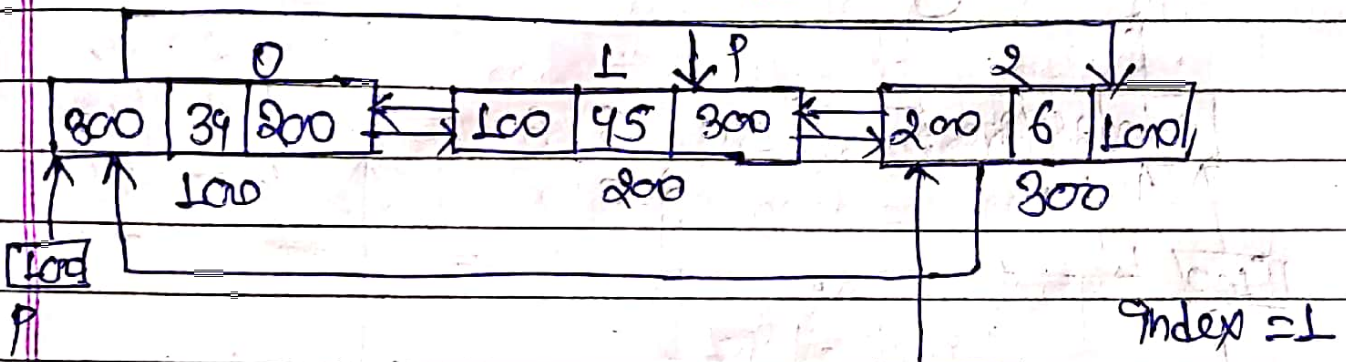
(11) Insertion at the end

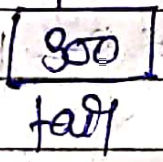


```
node *p = tail->next;
temp->next = tail->next;
temp->prev = tail;
 tail->next = temp;
 ~~tail->prev->prev = temp~~
 p->prev = temp;
  tail = temp;
```

O(1)

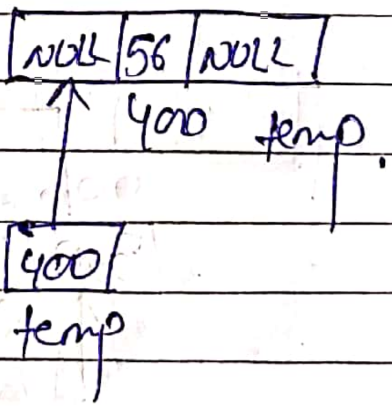
(iii) Insertion betⁿ two nodes





```
node *p = tail -> next;
 int i = 0;
 while ( i ! = index ){
            p = p -> next;
            i++;
         }
```

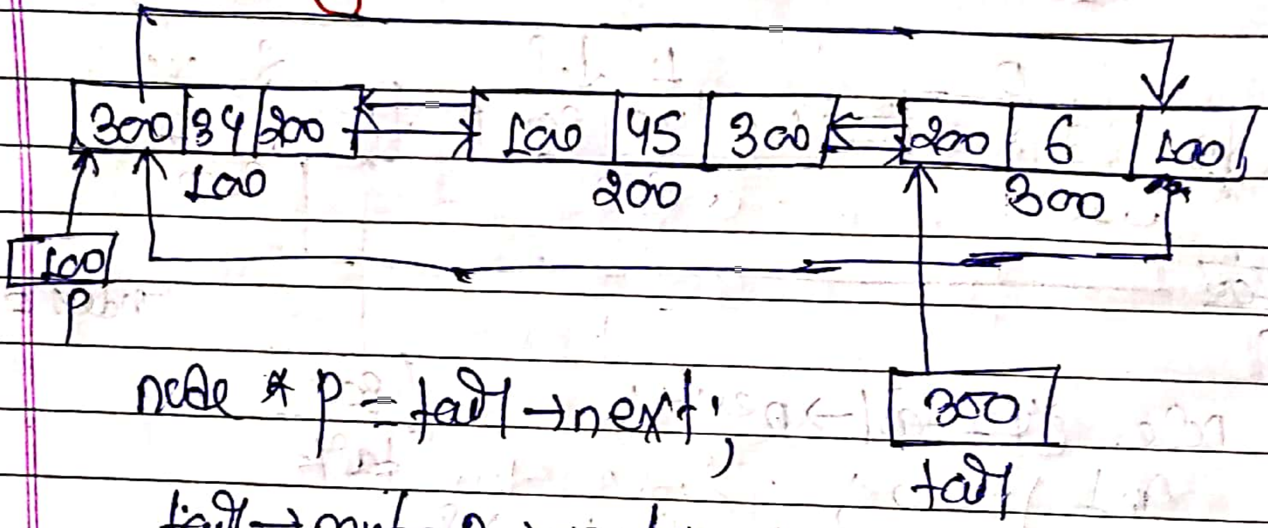
p ⇒ 200



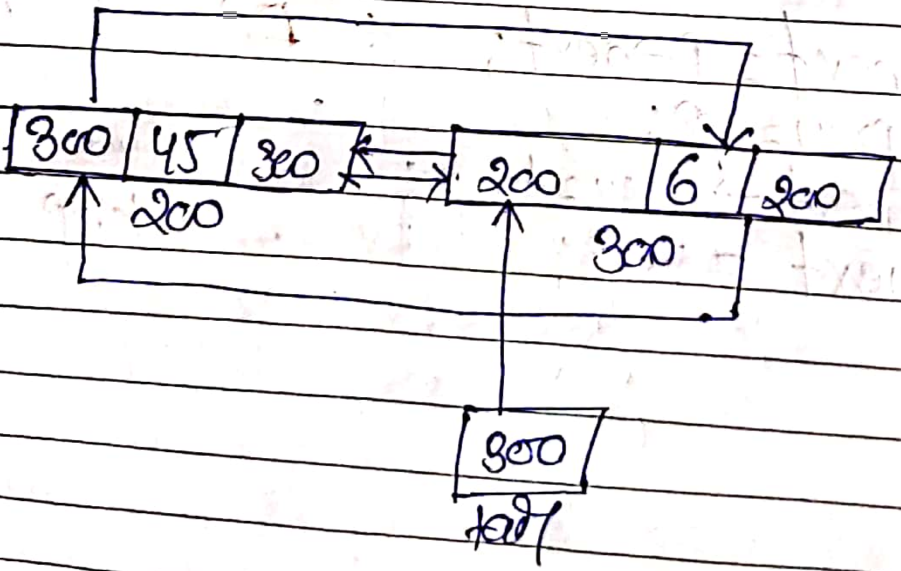
```
temp -> next = p -> next;
temp -> prev = p;
temp -> next -> prev = temp;
p -> next = temp;
```

$O(n)$

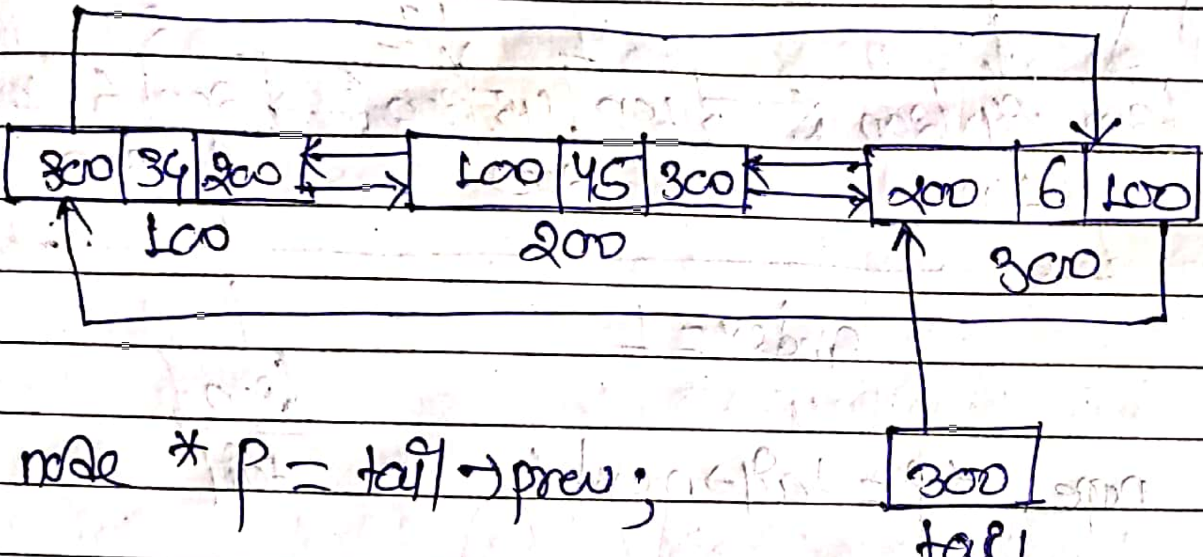
① Deleting first node



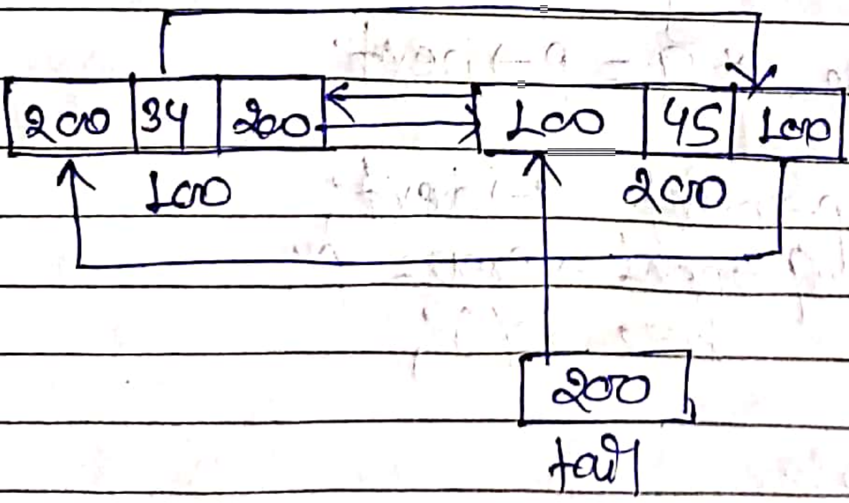
```
node *p = tail->next;
tail->next = p->next;
  p->next->prev = tail;
      free (p);
       p = NULL;
```



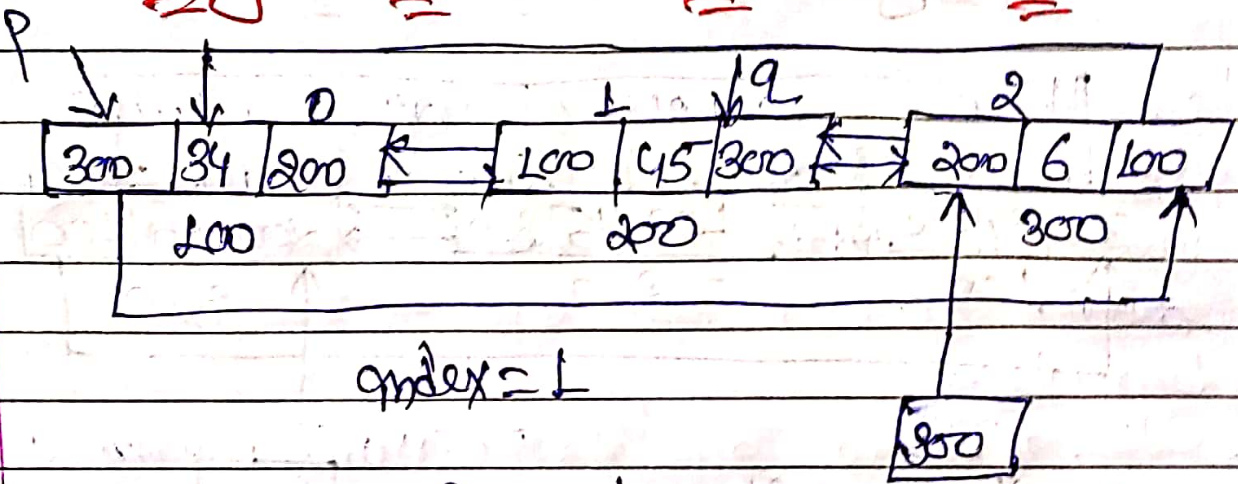
⑪ Deleting the last node



```
node *p = tail->prev;
    p->next = tail->next;
    tail->next->prev = p;
        free(tail)
        tail = p;
```



Deleting the Intermediate node



index = 1

```
node *p = tail->next;

    int i=0;
    while (i != index-1){
            p=p->next;
            i++;
    }
    node *q = p->next;

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