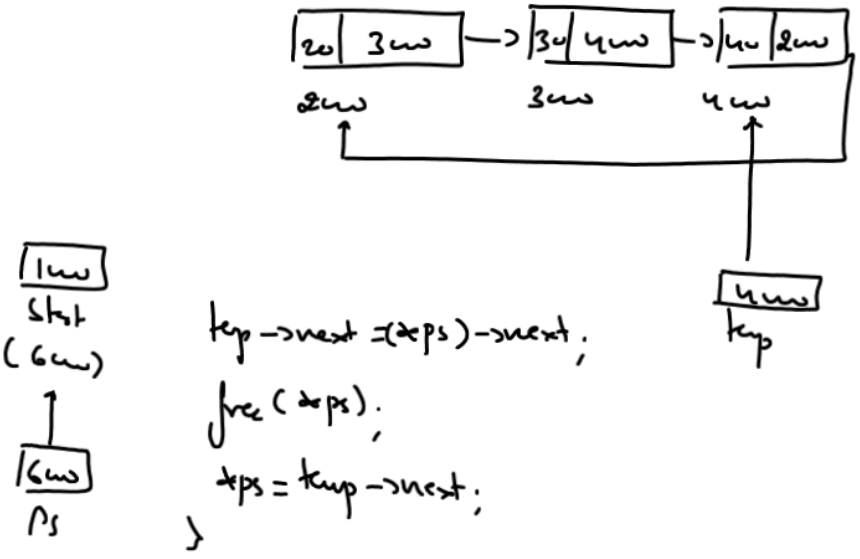
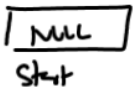
```
void del_first ( struct cnode **ps )
{
    struct cnode *temp;

    if( *ps == NULL)
    {
        printf("List is empty");
        return;
    }
    if( (*ps)->next == *ps)
    {
        free(*ps);
        *ps = NULL;
        return;
    }
    temp = *ps;

    while ( temp->next != *ps)
    {
        temp = temp->next;
    }

    temp->next = (*ps)->next;
    free(*ps);
    *ps = temp->next;
}
```



Deleting Last Node of Circular Linked List

Before Del

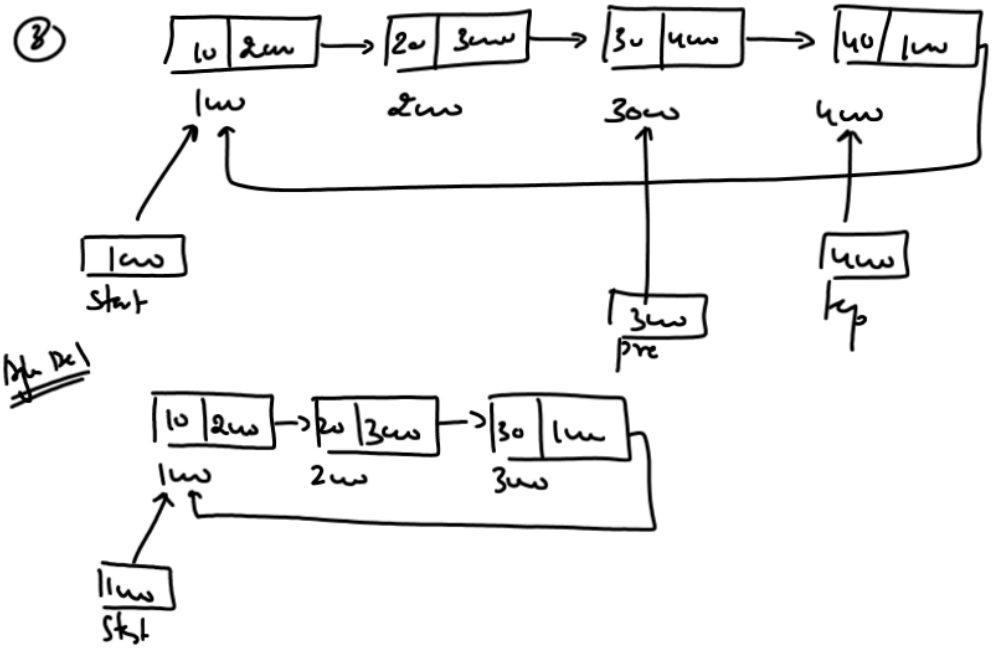
After Del

① 

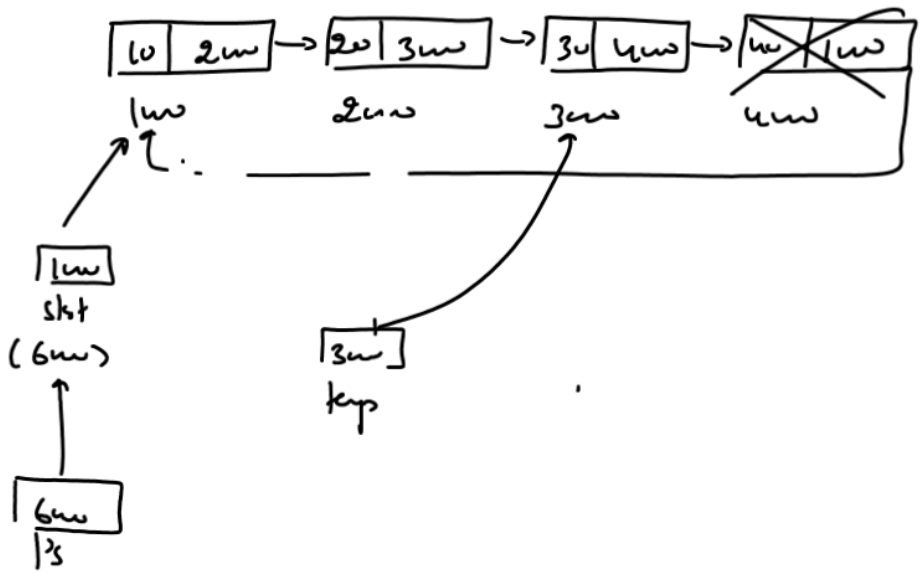
① List is empty

② 

② 

③ 

```
void del_last (struct node **eps)
{
    struct node *prev, *temp;

    if (*eps == NULL)
    {
        printf("List is empty");
        return;
    }

    if ((*eps)->next == *eps)
    {
        free(*eps);
        *eps = NULL;
        return;
    }

    temp = *eps;

    while (temp->next != *eps)
    {
        prev = temp;
        temp = temp->next;
    }

    free(temp);
    prev->next = *eps;
}
```



Solution 2

```
void del_last (struct cnode **eps)
{
    struct cnode *temp;

    if (*eps == NULL)
    {
        printf("List is empty");
        return;
    }
    if ((*eps)->next == *eps)
    {
        free(*eps);
        *eps = NULL;
        return;
    }
    temp = *eps;
    while (temp->next->next != *eps)
    {
        temp = temp->next;
    }
    free(temp->next);
    temp->next = *eps;
}
```

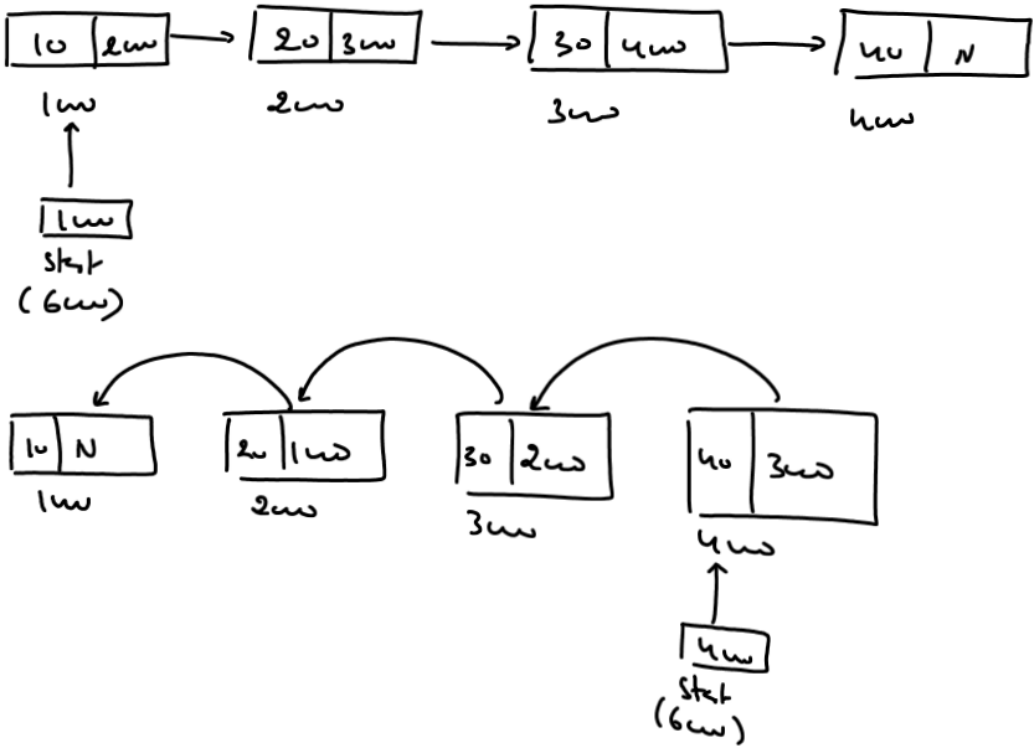
ASSIGNMENTS
==========

Qn1. WAF called **del_any()** which should accept an **int** as argument and deletes that node from **CIRCULAR LINKED LIST** whose data part matches with the arguemnt passed.

Qn2. WAF called **insert()** which should add a new node in a CIRCULAR LINKED LISt in such a way that the list always remains sorted.

Qn3. WAF called **print_reverse()** whic displays a LINEAR LINKED LIST in reverse order.

Qn4. WAF called **reverse()** which REVERSES a LINEAR LINKED LIST



Doubly Linked List