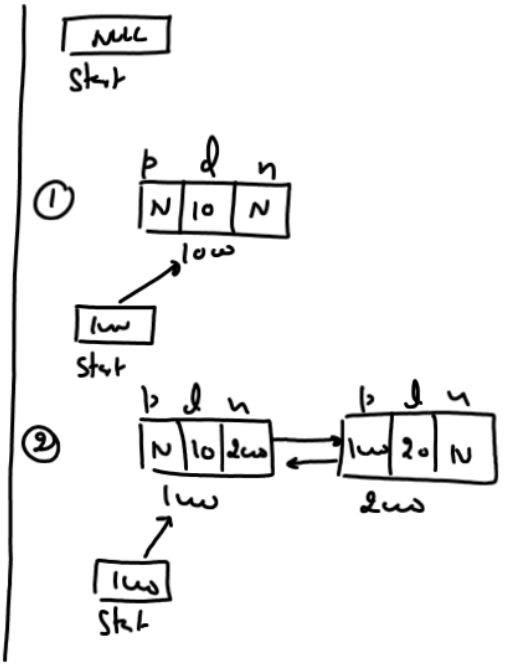
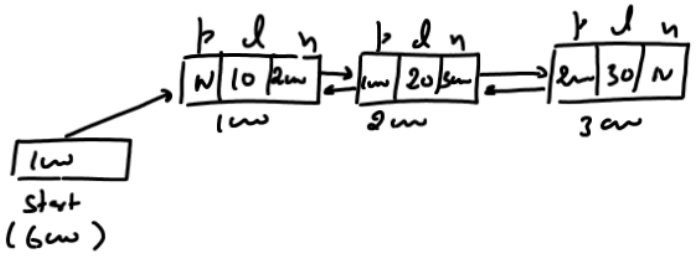
Implementing Doubly Linked List

```
struct doubly
{
   struct doubly *prev;
   int data;
   struct doubly *next;
};
void append(struct doubly **,int);
void display(struct doubly *);
int main()
{
   struct doubly *start=NULL;
   append(&start,10);
   append(&start,20);
   ....
   display(start);
    return 0;
}
```



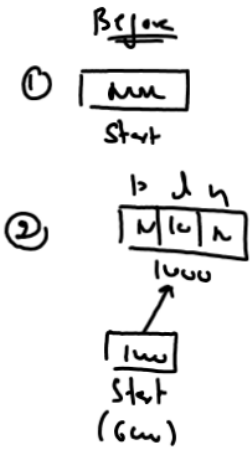

```
void append(struct doubly **ps,int x)
{
   struct doubly *p,*temp;
   p=(struct doubly *) malloc(sizeof(struct doubly));
   if(p==NULL)
    {
        printf("Insufficient memory");
        return;
    }
p->data=x;
p->next=NULL;
if(*ps==NULL)
  {
     p->prev=NULL;
     *ps=p;
     return;
  }
temp=*ps;
while(temp->next!=NULL)
{
   temp=temp->next;
}
temp->next=p;
p->prev=temp;
}
```

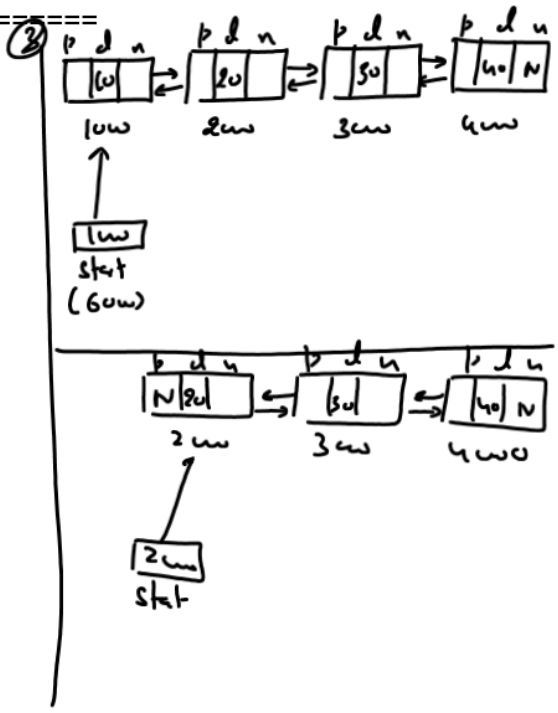


```
void display(struct doubly *p)
{
    if(p==NULL)
   {
      printf("List is empty");
      return;
  }
   while(p!=NULL)
   {
       printf("\n%d",p->data);
       p=p->next;
   }
}
```

Deletion In Doubly Linked List
==========

Deleting FIRST NODE
==========

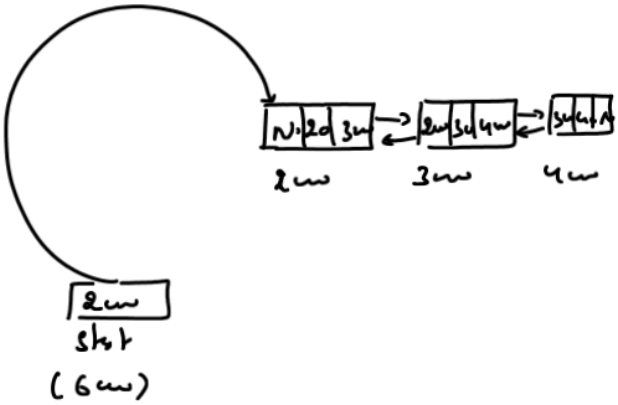




```
void del_first(struct doubly **ps)
{
   struct doubly *temp;

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

if((*ps)->next==NULL)
   {
        free(*ps);
        *ps=NULL;
        return;
   }
*ps=temp->next;
(*ps)->prev=NULL;
free(temp);
}
```

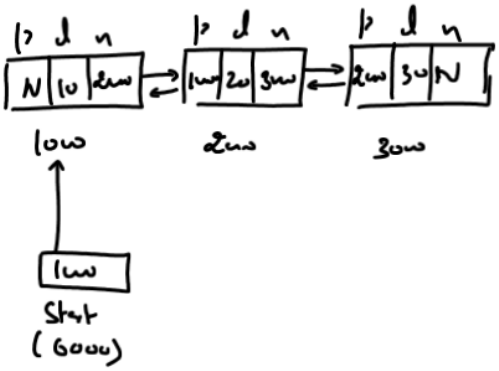


Deleting Last Node
==========

```
void del_last(struct doubly **ps)
{
  struct doubly *temp;

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

if((*ps)->next==NULL)
   {
        free(*ps);
        *ps=NULL;
        return;
   }
   while(temp->next!=NULL)
   {
       temp=temp->next;
   }
   temp->prev->next=NULL;
   free(temp);
}
```



WAF called del_any() which accepts an int as argument and searches and deltes that node from the list whose data part matches with the argument passed.