unacademy

# Linked List - Part VI

Course on Data Structure

Pankaj Sharma • Lesson 12 • July 22, 2024

unacademy

# CS & IT Engineering

## Data Structure

### Linked List

Lecture Number- 12

By- Pankaj Sir

# Topics

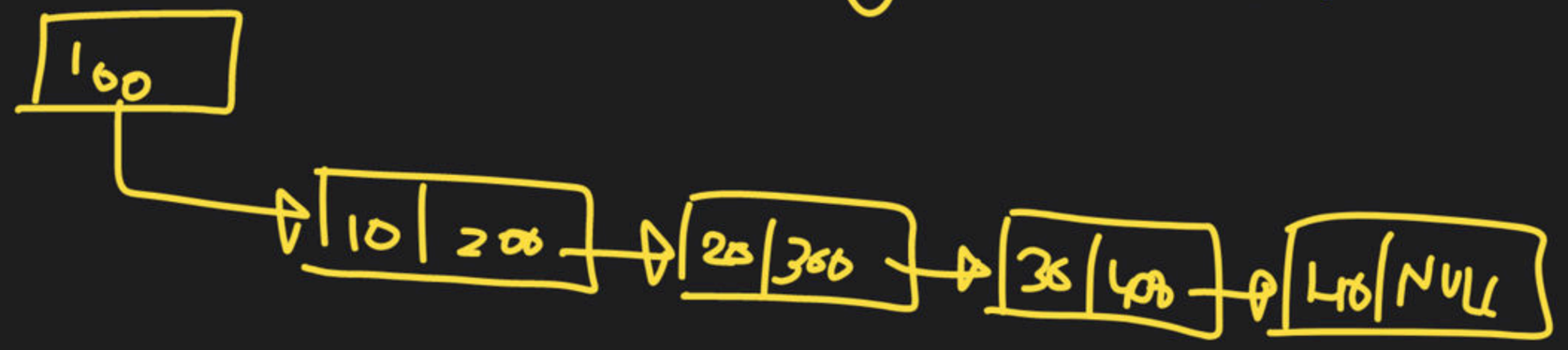
*to be covered*

1. Linked List

Begin

START → global variable

START

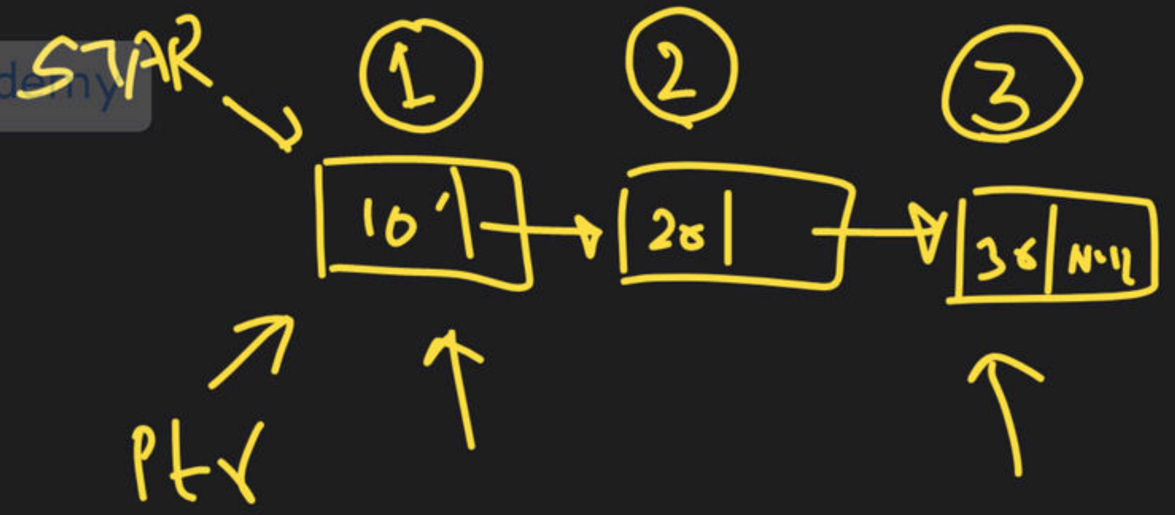


unacademy

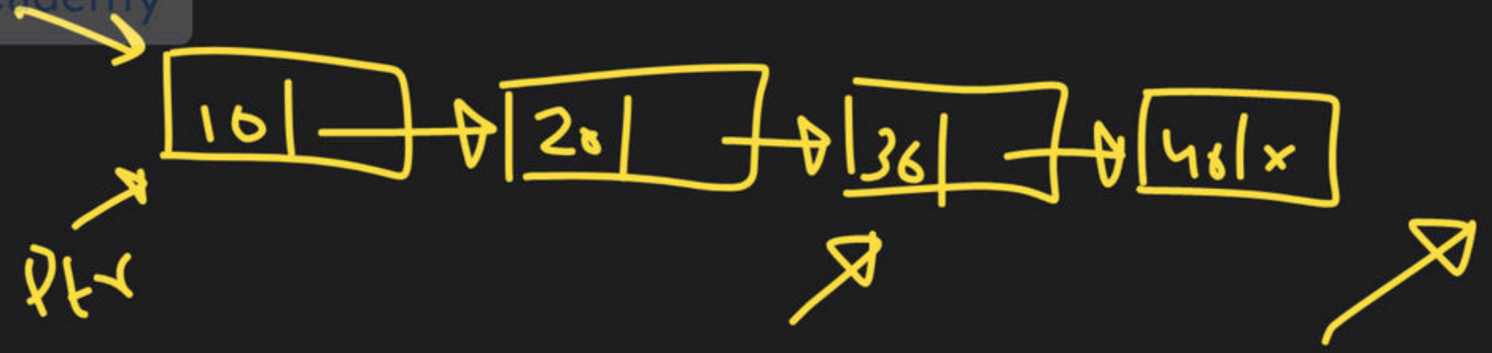
▲ 2 • Asked by Adarsh

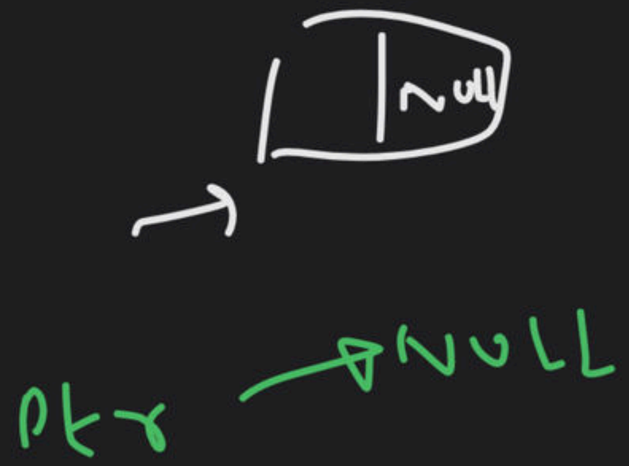
sir ll ke alternate node ko print kerne vala code plese ek bar

.




```
int c=1;

while(Ptr!=NULL)
{
    if(c%2==1)
        pf("%d",Ptr->data);
    Ptr=Ptr->Next;
}
```



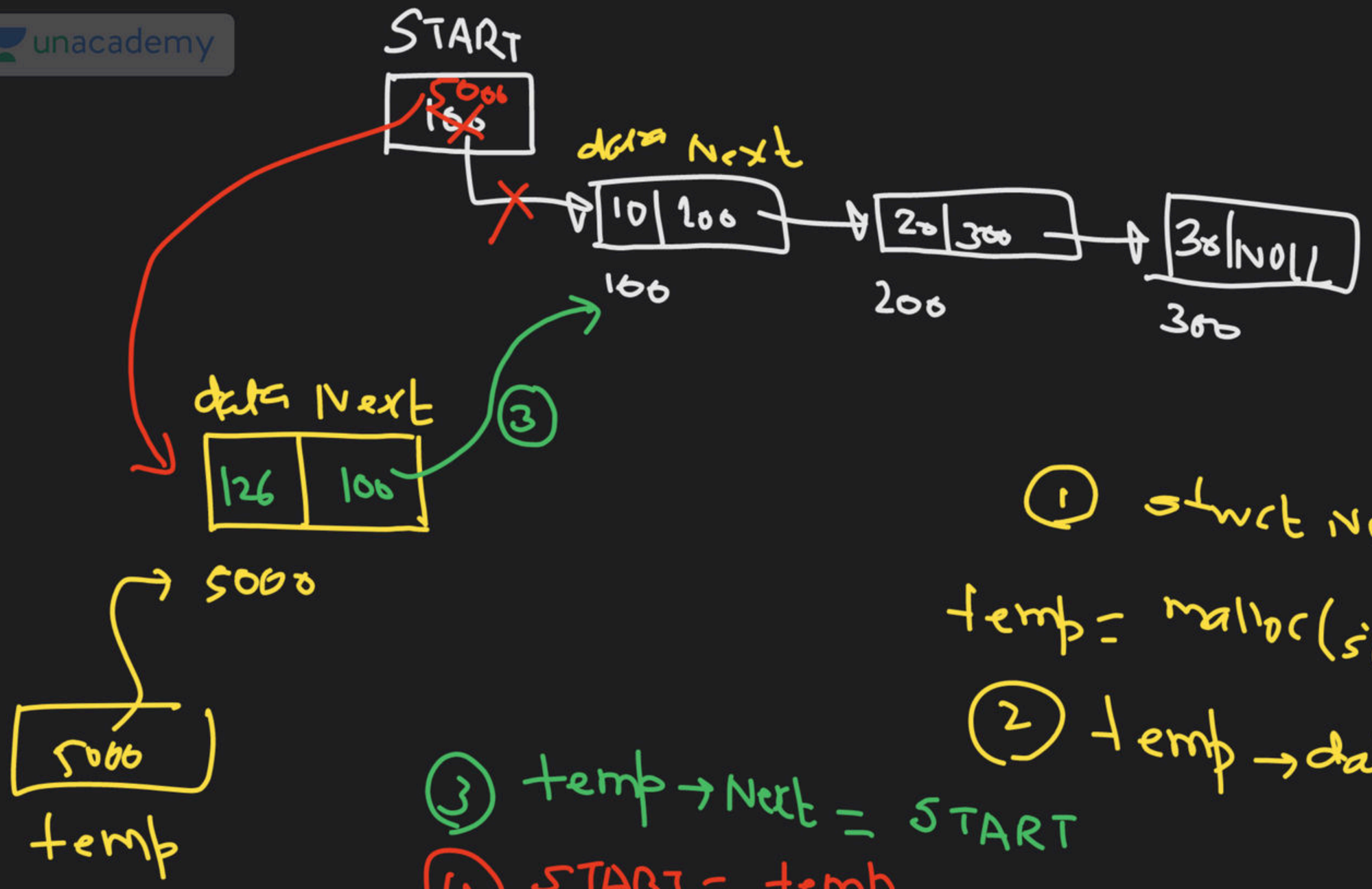

False

```
while (Ptr != NULL && Ptr -> Next != NULL)
{
    pf("%d", Ptr -> data);
    Ptr = Ptr -> next -> next;
}
```

unacademy

logic

```
while (Ptr != NULL && Ptr -> Next != NULL)
{

}
if (Ptr == NULL) return;
if (Ptr -> Next == NULL)
        Pf("%d", Ptr -> data);
```




Key = 126

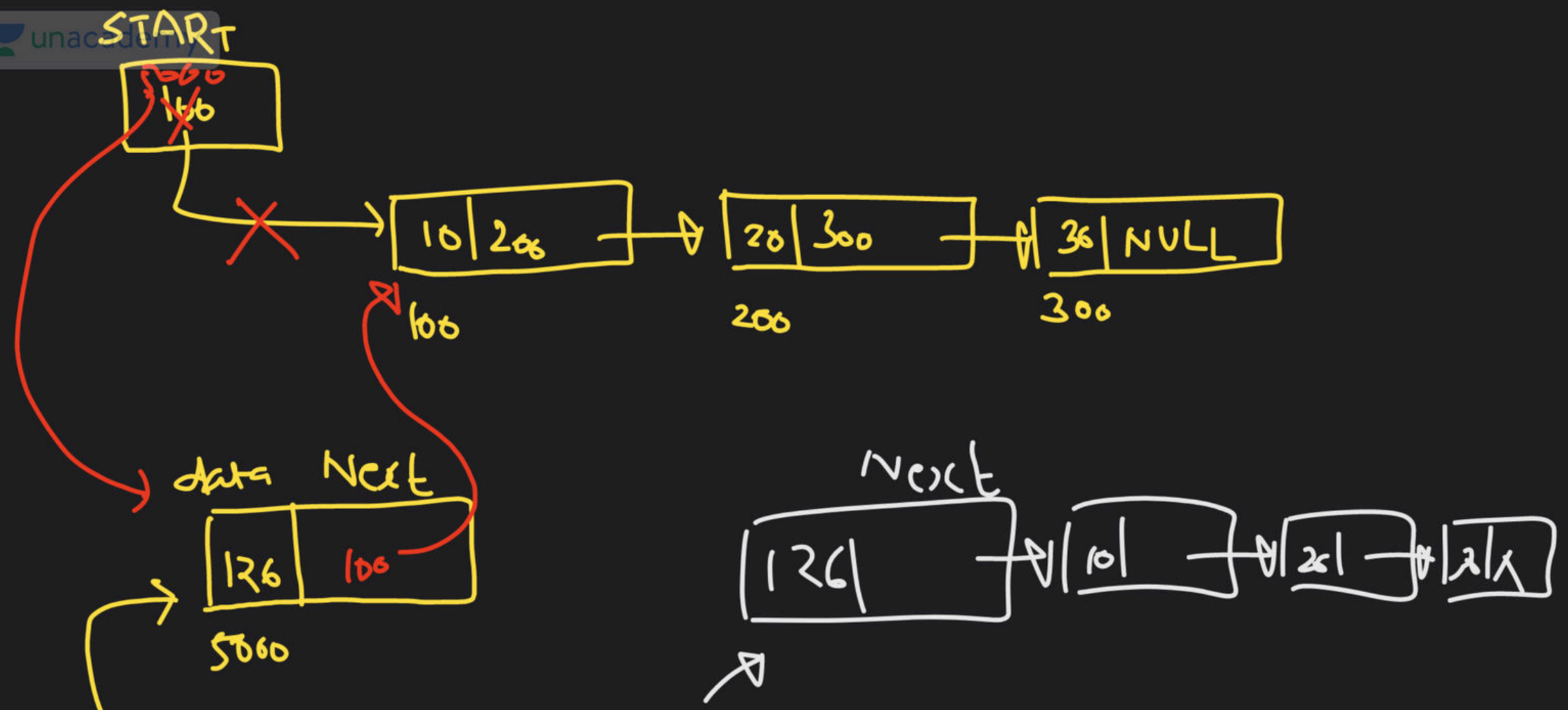
① struct Node * temp;

temp = malloc(sizeof(struct Node

② temp → data = key

③ temp → Next = START

④ START = temp




temp->Next = START;

START = temp;

```
void Insert_at_begin(int key){
    struct Node *temp;
    temp = malloc(sizeof(struct Node));
    if(temp!=NULL){
        temp->data = key;
        temp->Next = START;
        START = temp; }
    }
```

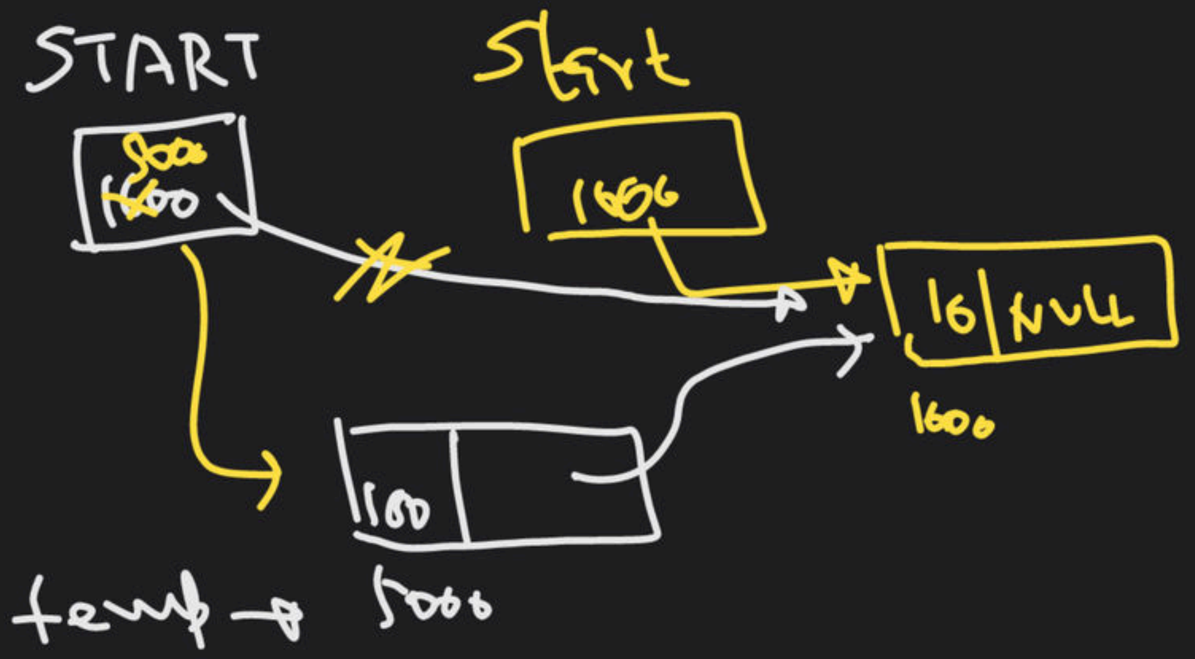
```
void main(){
    ==
    ==
    Insert(key);
    ==
    ==
    }
```

```
void Insert_at_begin(int key){
  struct Node *temp;
  temp = malloc(sizeof(struct Node));
  if(temp!=NULL){
      temp->data = key;
      temp->Next = START;
        START = temp; }
  }
```

```
void Insert_at_begin(struct Node
                     *START,
                     int Key)
{
```

```
void main(){
struct Node *Start;
 ≡
Insert_at_begin(Start,100)
 ≡
}
```

START=temp;

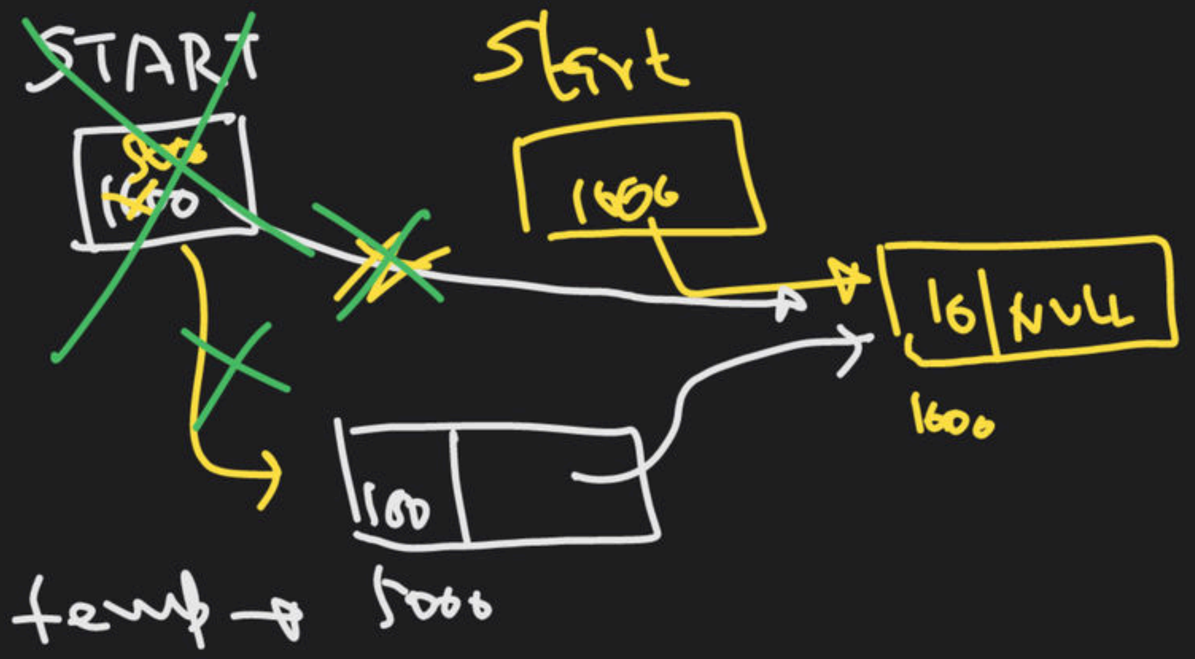



}

```
void Insert_at_begin(struct Node
                     *START,
                     int Key)
{
```

START=temp;

}



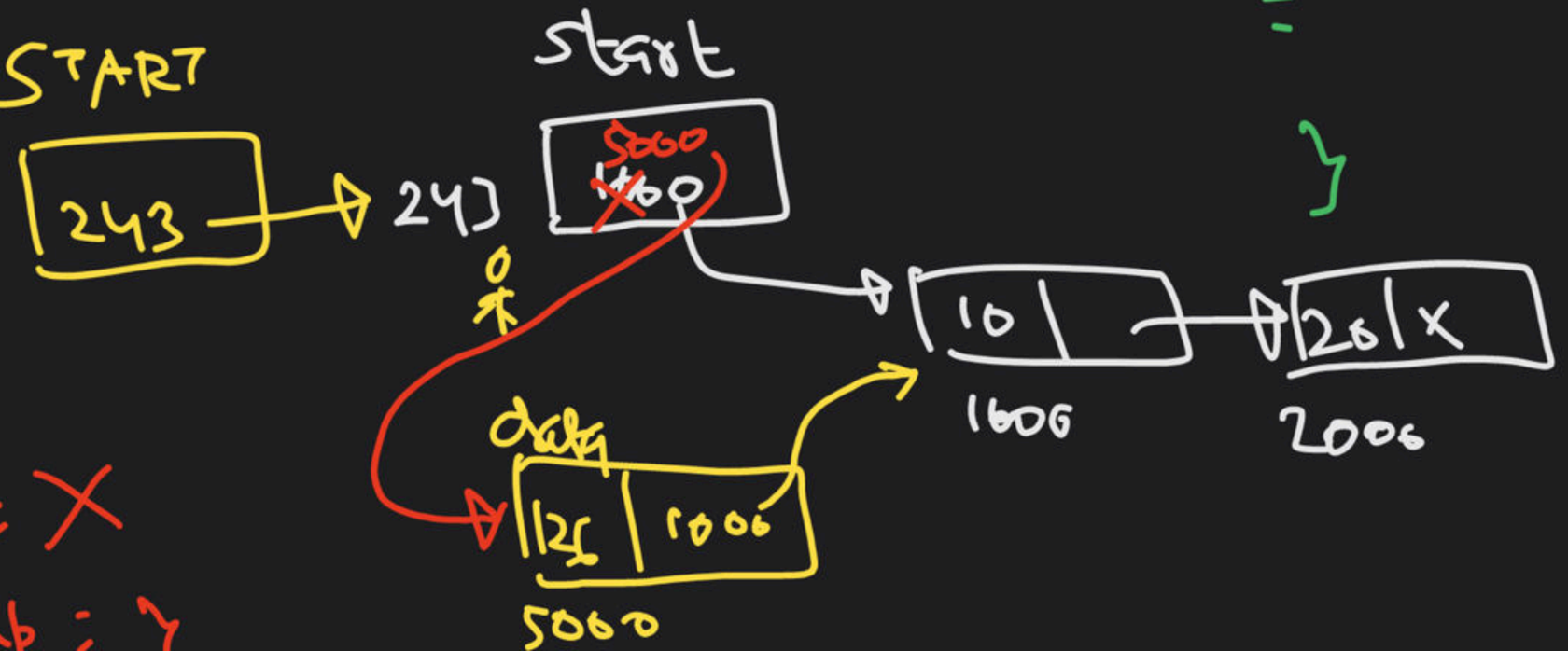

```
void main(){
struct Node *Start;
=
Insert_at_begin(Start,100)
=
}
```

1606

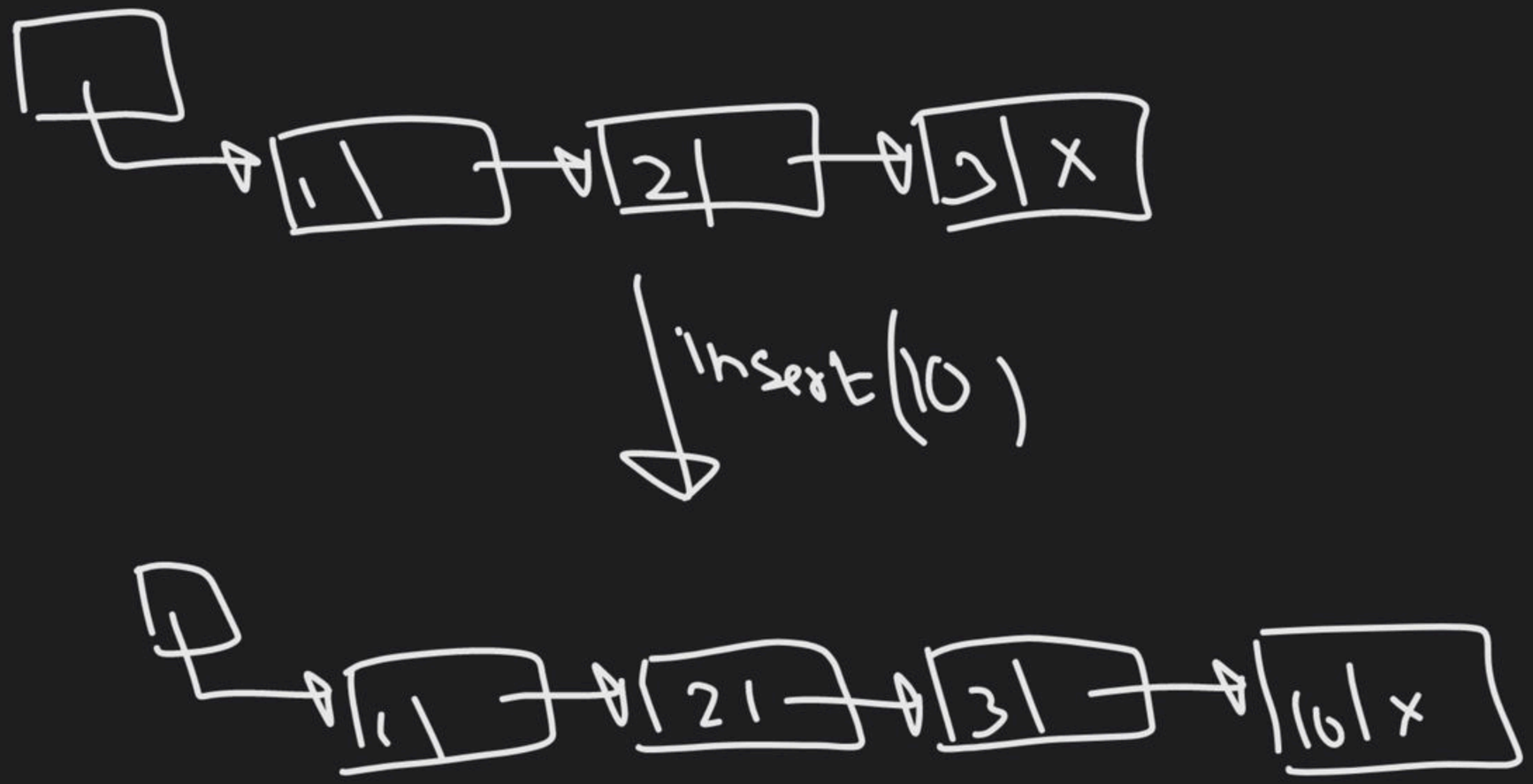
call by value

```
void insert_at_begin(struct Node **START,
                     int key)
{
  struct Node *temp;
  temp = malloc( ____ );

  temp->data = key

  temp->next
        = *START
  START = temp ; X
  *START = temp ; ✓
```

```
Void main(){
  struct Node *start;
  ⋮
  Insert_at_begin(&start, 126)
  ⋮
}
```



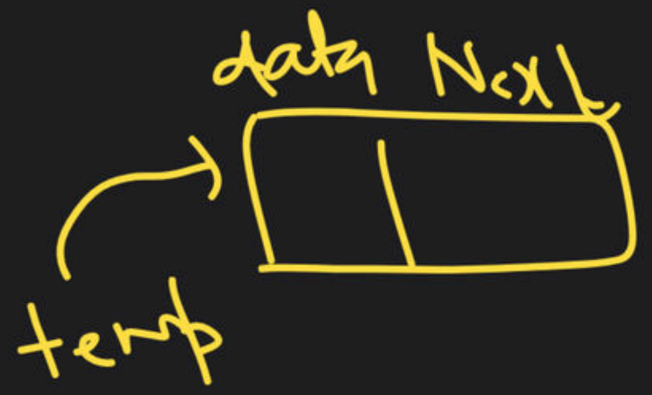
# Insert at End (START is global)



① Allocate

```
struct Node *temp;
temp = malloc(sizeof(struct Node));
```

②

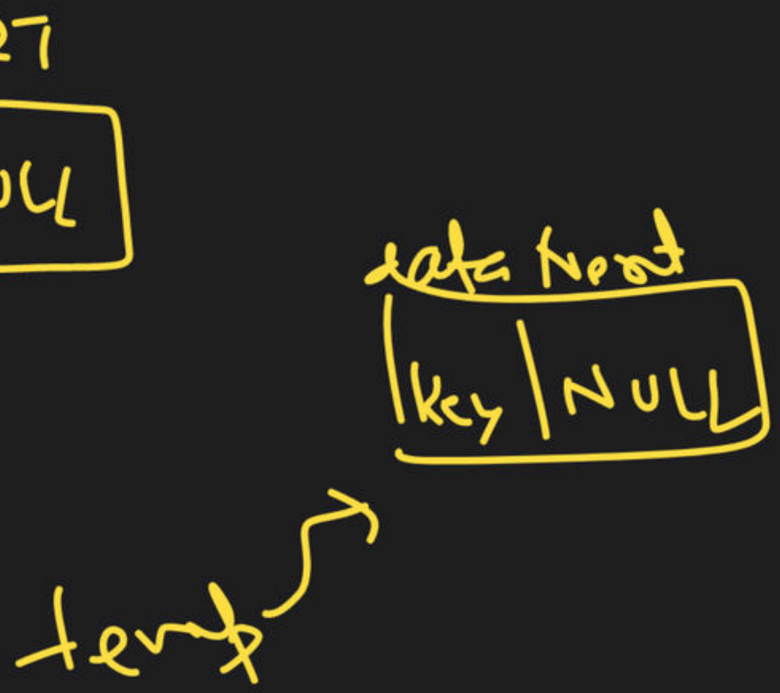



```
temp->data = Key;
temp->Next = NULL;
```

Case 1. LL is Empty

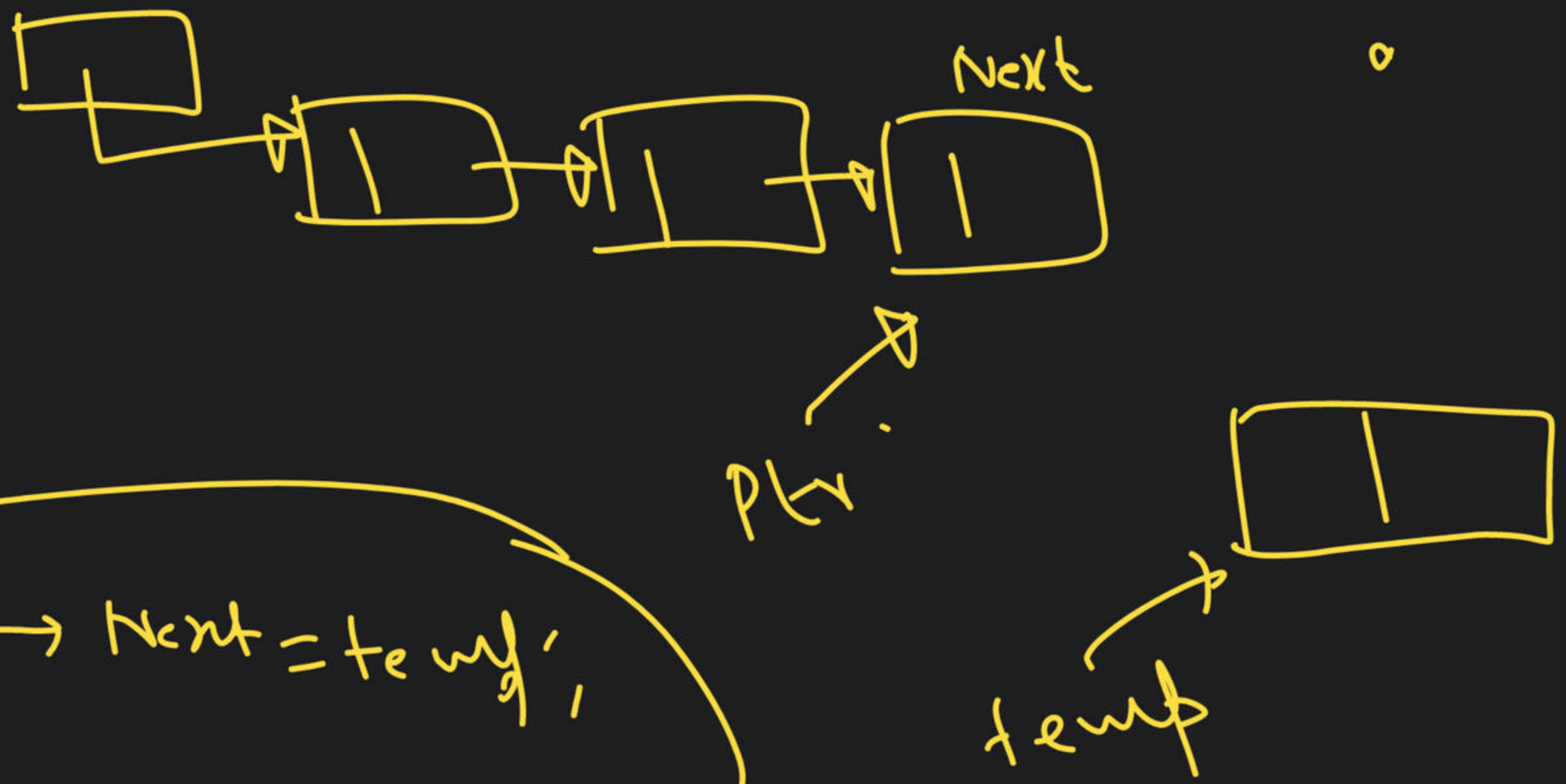
START = temp;

return;



LL is not Empty

reach last Node → ?




Ptr → Next = temp;

Isse table
ensure karlo
ki LL
empty nihai

```
struct node *ptr;
Ptr = START;
while(Ptr->Next != NULL)
    Ptr = Ptr->Next;
```

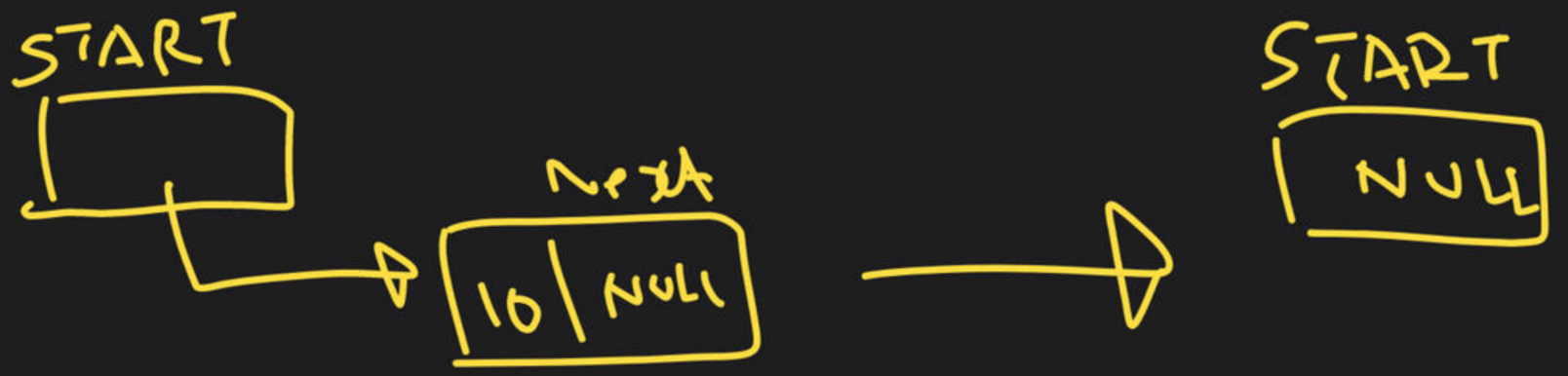
```
void Insert_at_End(int key){
    struct Node *temp, *Ptr;

    temp = malloc(sizeof(struct Node));
    if (temp != NULL){
        temp->data = key;
        temp->Next = NULL;

        if (START == NULL)
        {
            START = temp;
            return;
        }
        Ptr = START;
        while(Ptr->Next != NULL)
            Ptr = Ptr->Next;
        Ptr->Next = temp;
    }
}
```

# Deletion from a linked list

Begin

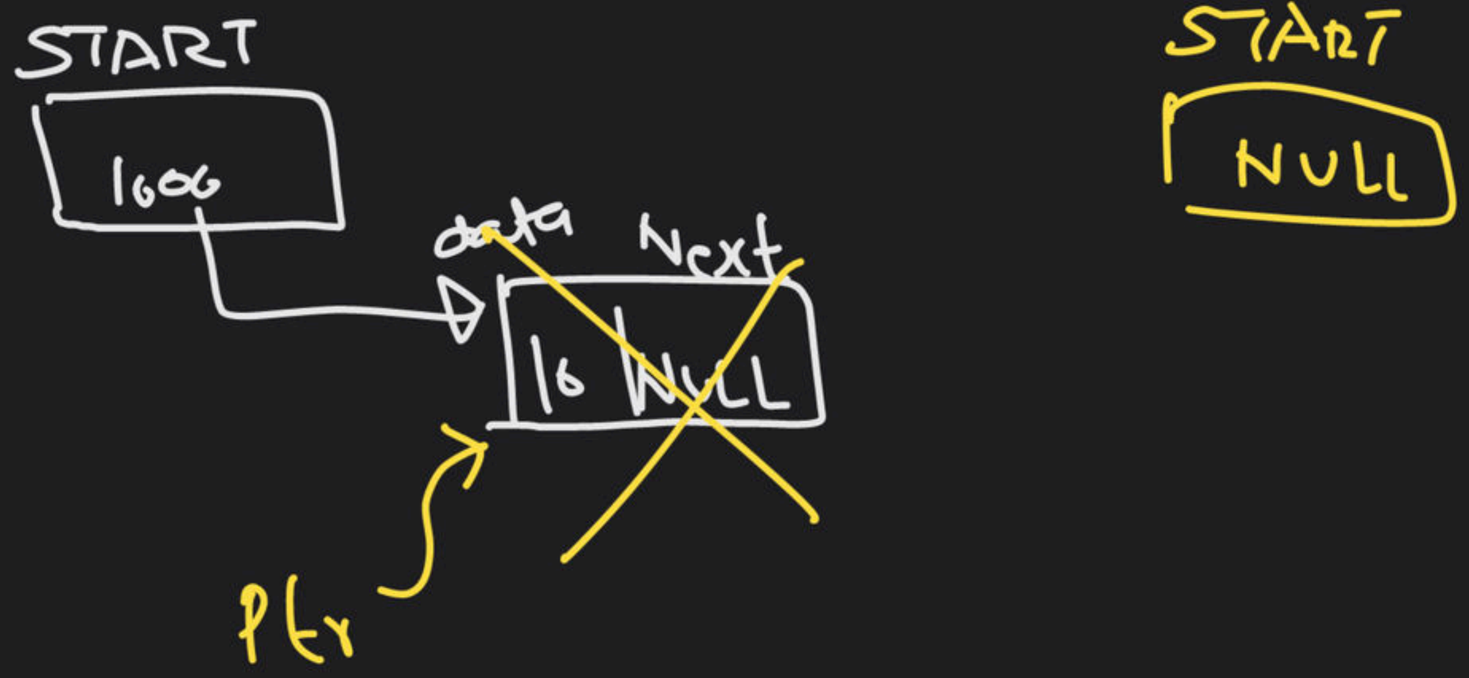
Case 1:


Case 2

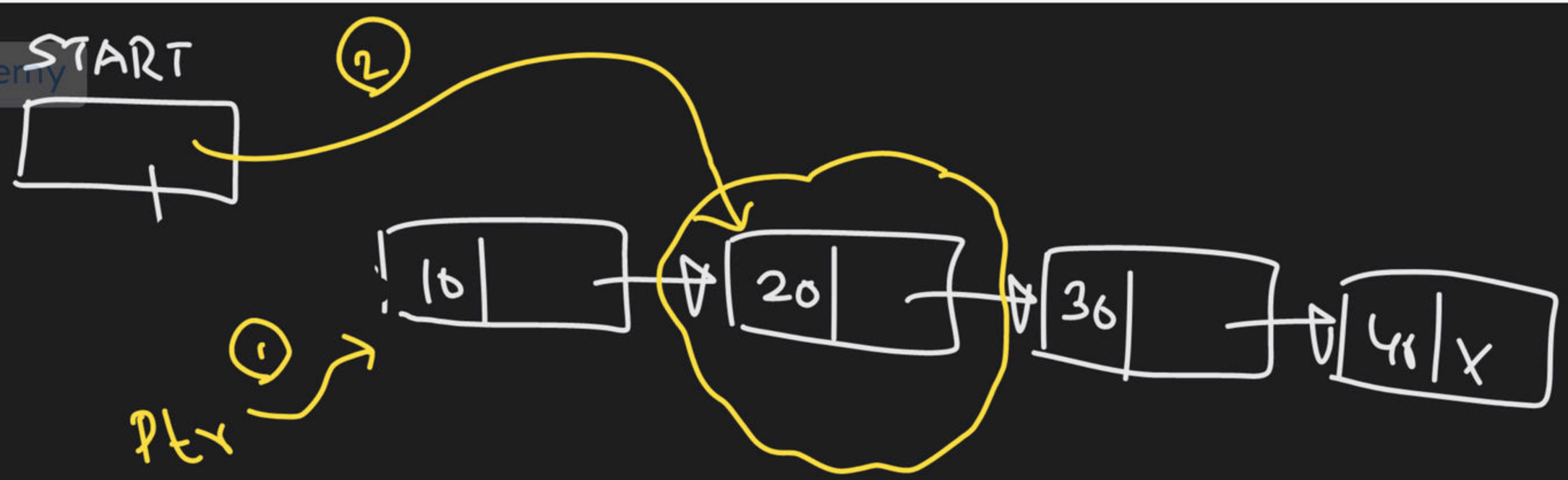



DMA

Case 2.




```
struct Node *ptr;
    Ptr = START
START = NULL
    free(ptr)
```

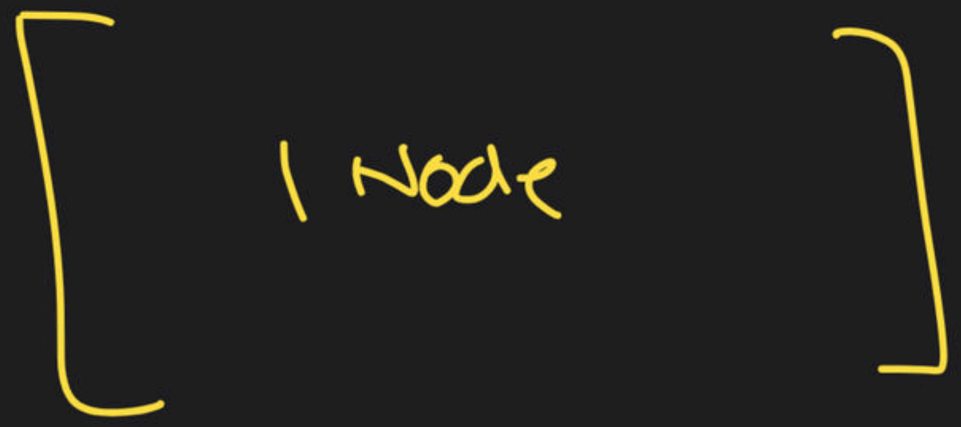



```
struct Node *Ptr = START
START = START -> Next;
free(Ptr);
```

After deletion this will become 1st node

Q) case 2 → case 3 (case 3 will handle case 2 or not)

```
void delete() {
struct Node *ptr;
 if(START == NULL)
        return;
```

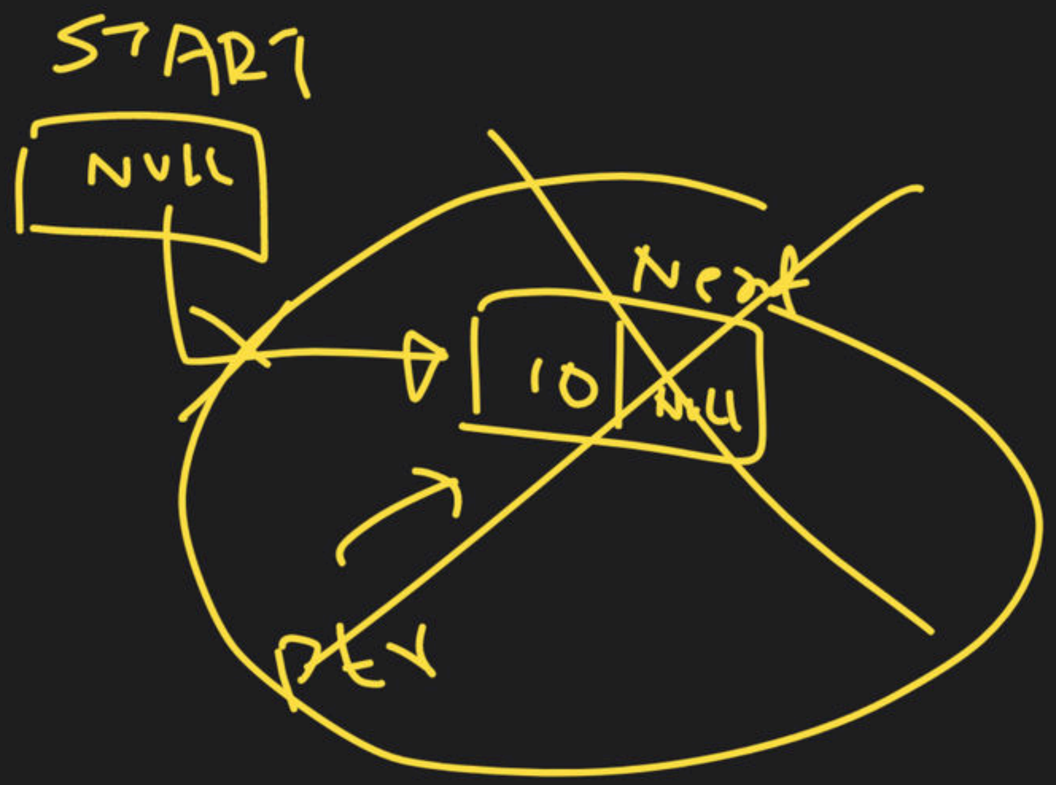



```
    ptr = START;
  START = START->Next;
      free(ptr);
```

```
void delete() {
struct Node *ptr;
 if(START == NULL)
        return;
```

[ 1 Node ] No need

```
    ptr = START;
  START = START->Next;
      free(ptr);
```



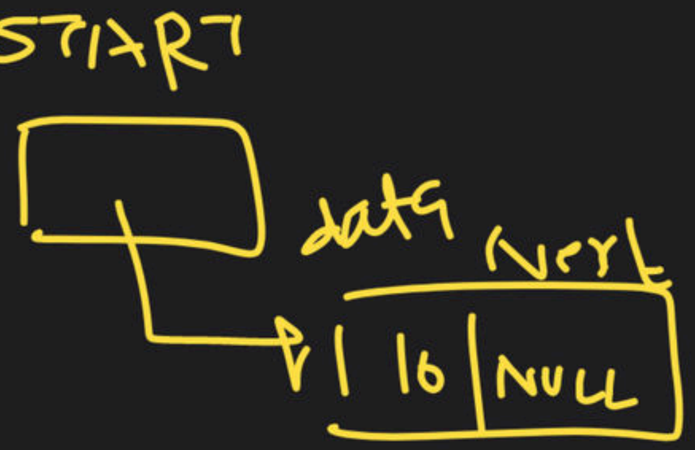
```
void delete_begin(){
  struct Node *ptr;
  If(START==NULL)
      return;
  Ptr = START;
  START = START→Next;
      free(ptr);
}
```

# Deletion of last Node

**Case 1**

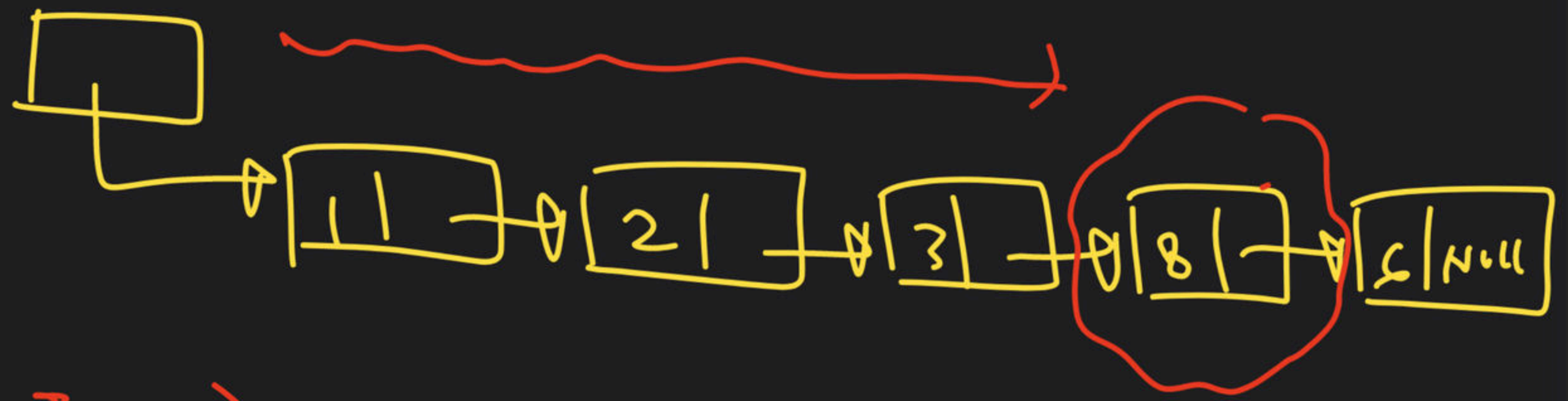

Do nothing ⇒ return

**Case 2**




⇒
```
{ Ptr = START;
  START = NULL;
  free(Ptr) }
```

Atleast 2 nodes




Insert
delete
search
Traversal
count Nodes

After deletion this will becomes last

⇓

we have to set NULL in its next field

unacademy

# THANK YOU!

Here's to a cracking journey ahead!