CS & IT ENGINEERING

Data Structure

Linked List
Chapter- 3
Lec- 03

By- Pankaj Sharma sir

Linked List-III

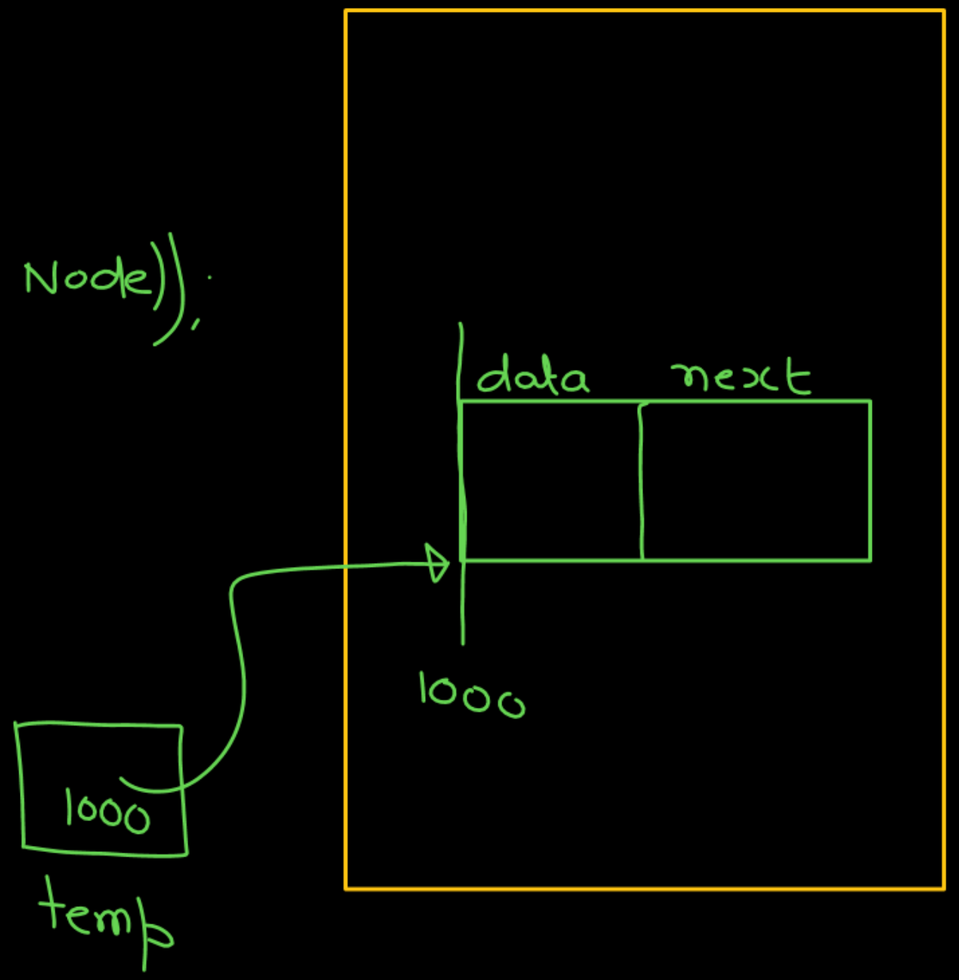
Insertion

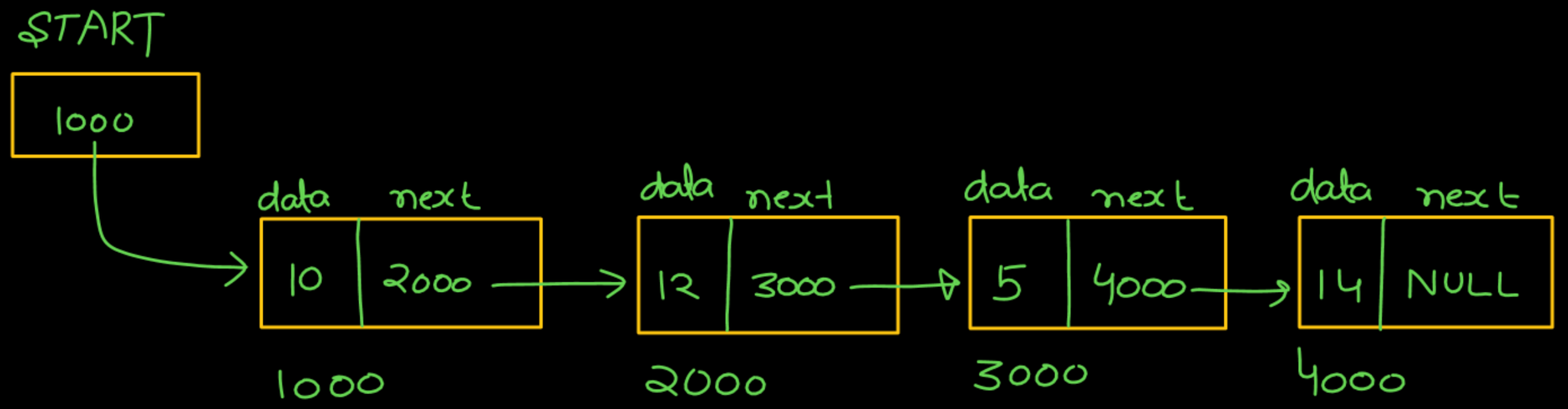
① **Memory allocate**

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

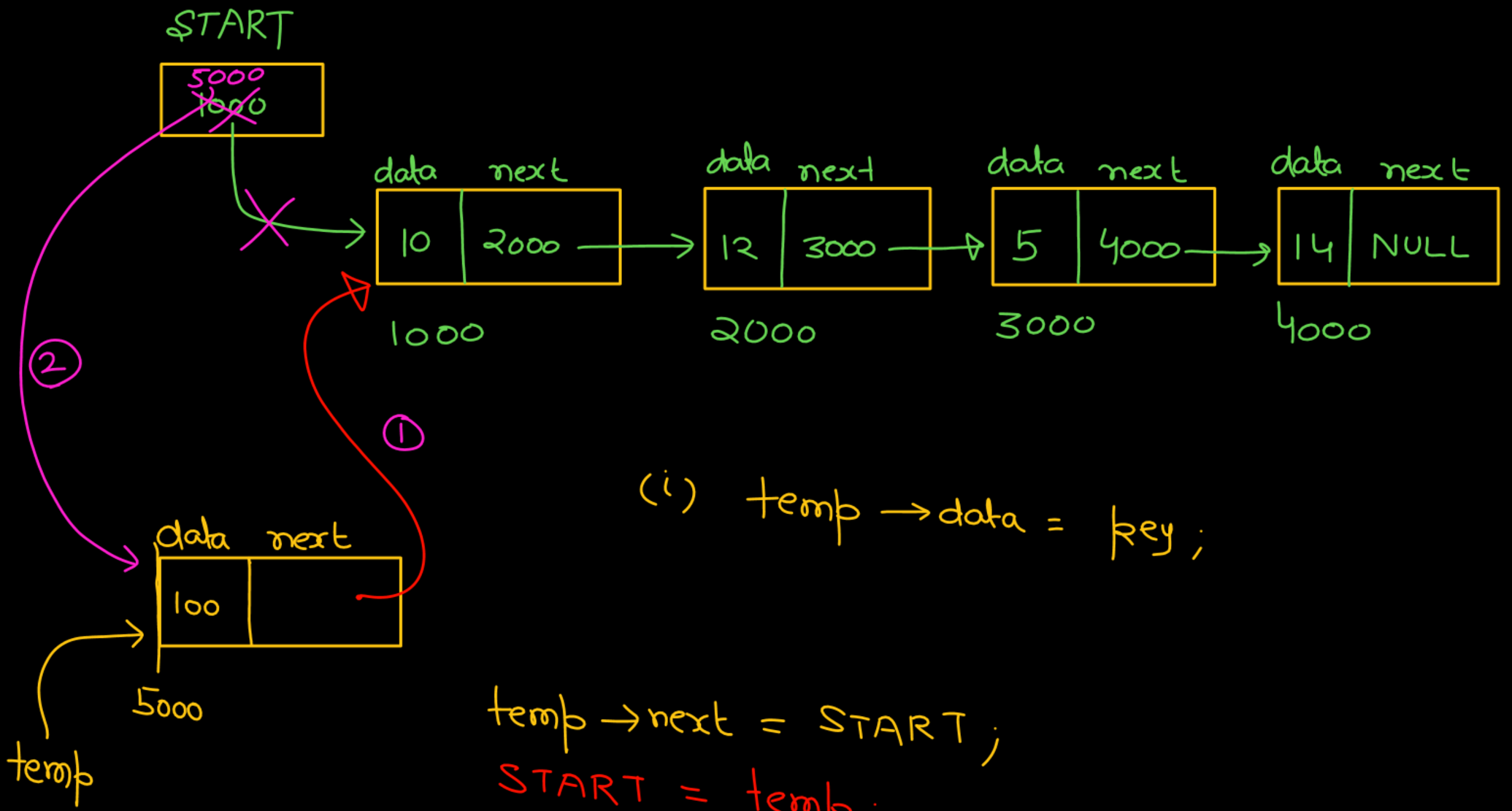
② a) Insert ⇒ Key

b) Where to insert





1) begining
2) End
3) After a given node



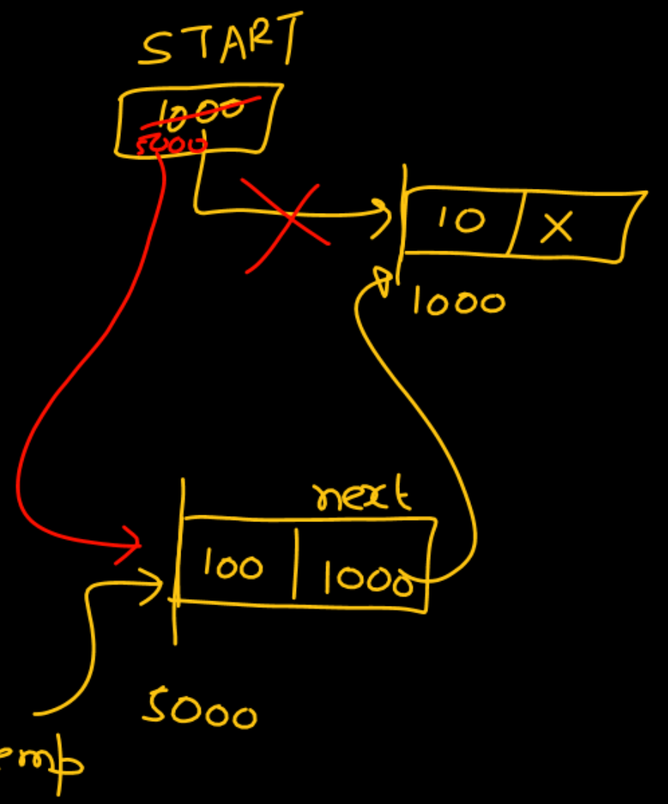
(i) temp → data = key ;

temp → next = START ;

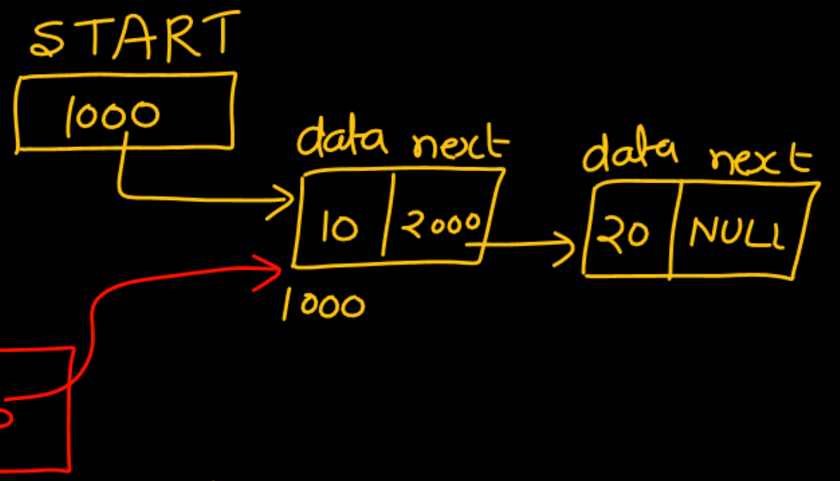
START = temp ;

START → global

```
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;
    }

}
```

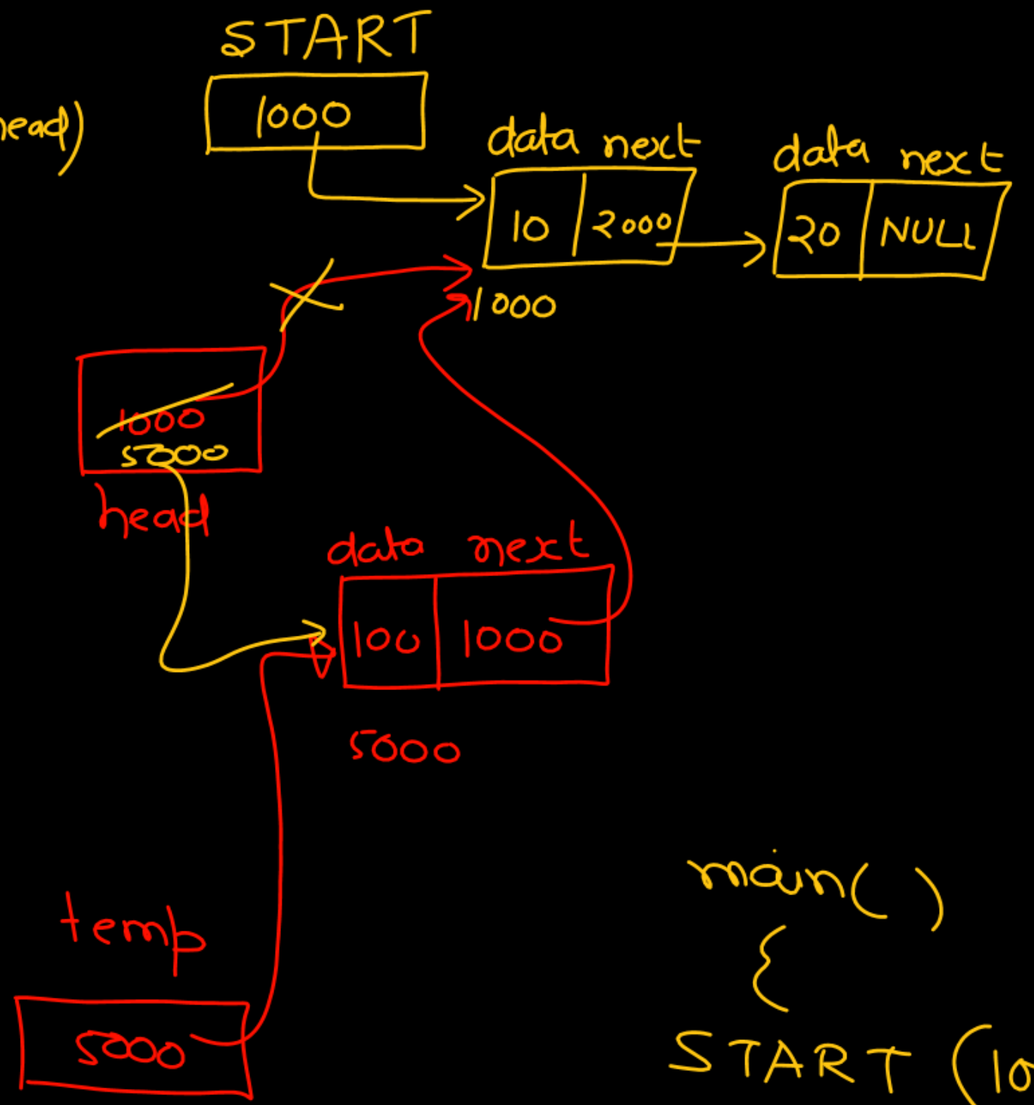


```
void Insert_warpan (int key, struct Node *head)
    {
    struct Node *temp;
  temp = malloc(sizeof(struct Node));
     if(temp != NULL)
      {
       temp->data = key;
       temp->next = head;
        head = temp;
      }

    }
```



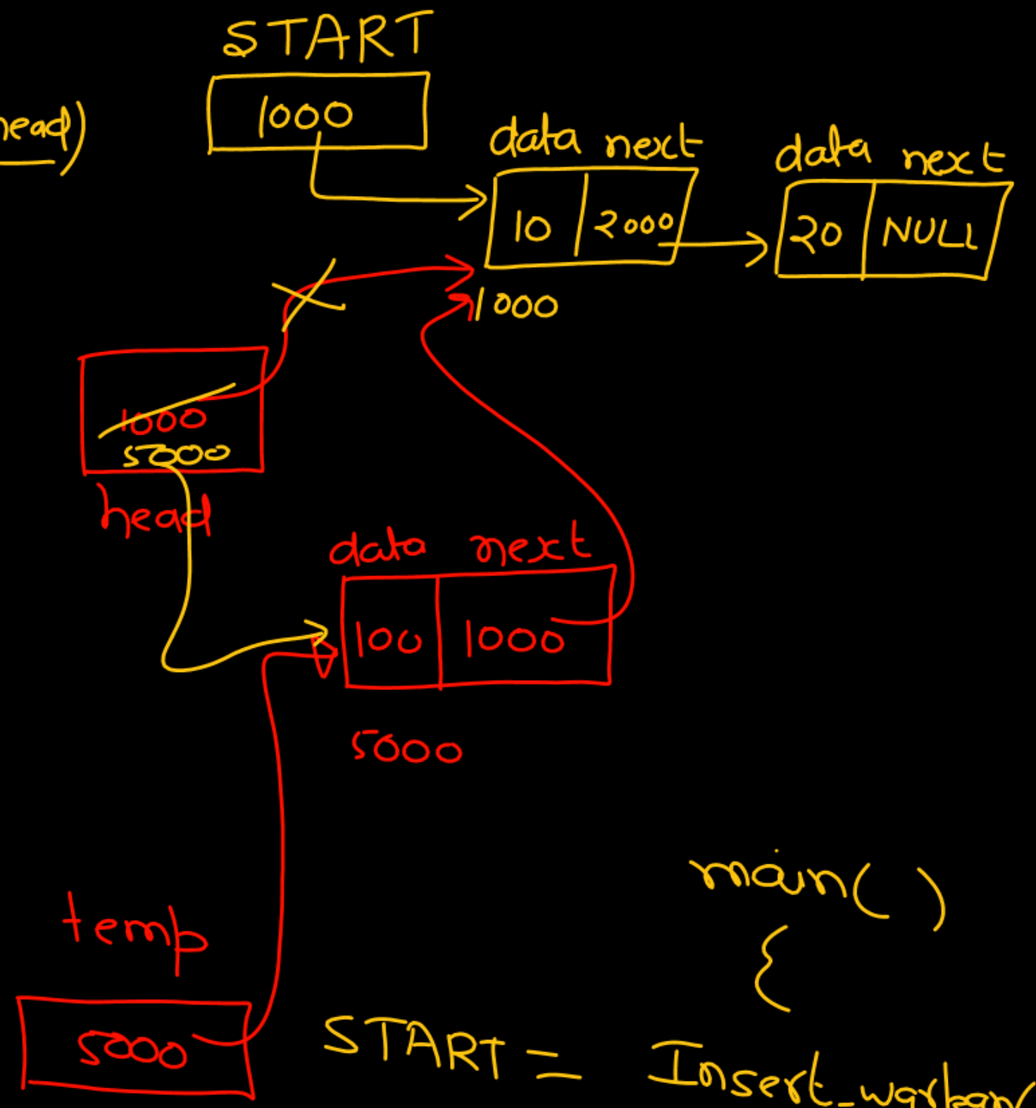

```
void main(){
struct Node *START=NULL;
   ||
   ||
Insert_at_beg(100, START);
   ||
   }
```

```
void Insert_warpan (int key, struct Node *head)
    {
    struct Node *temp;
  temp = malloc(sizeof(struct Node));
     if(temp != NULL)
      {
       temp->data = key;
       temp->next = head;
        head = temp;
      }

}
```



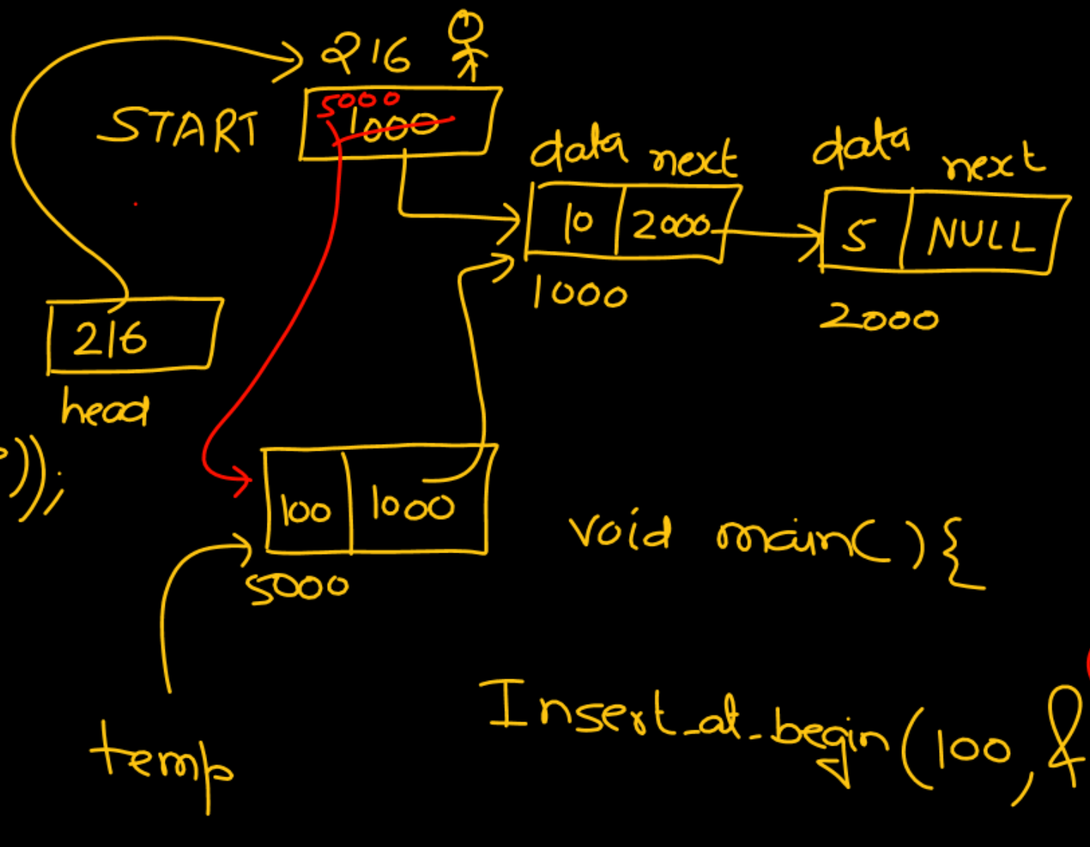

```
main( )
  {
START (1000)

}
```

```
void Insert_warkan (int key, struct Node *head)
    {
    struct Node *temp;
  temp = malloc(sizeof(struct Node));
     if(temp != NULL)
      {
       temp->data = key;
       temp->next = head;
        head = temp;
      }
    return head;
    }
```




```
main( )
  {
START = Insert_warkan(100,
                      START);
  }
```

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

}
```




```
void main(){

Insert_at_begin(100, &START);
```

216

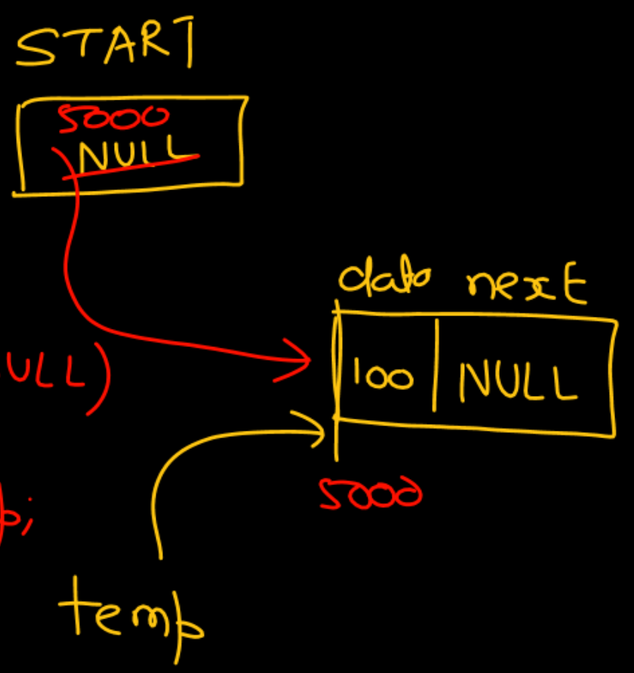
Insertion at End → last node of next ⇒ NULL

case1: Linked list is Empty

1] Allocate memory

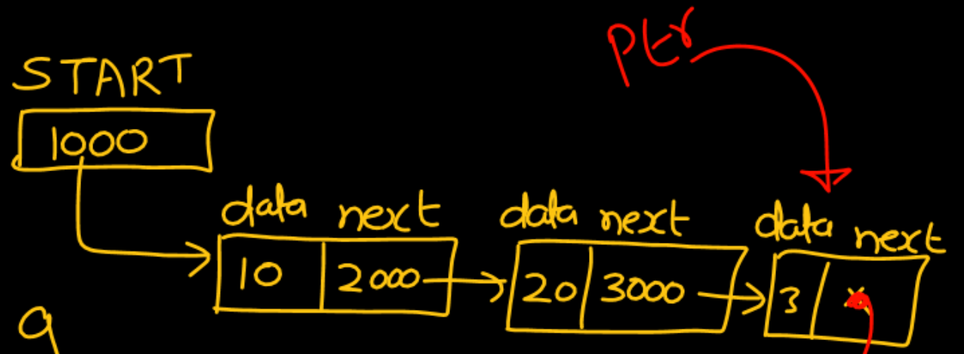
a] temp → data = key ;

b] temp → next = NULL ;



Non-Empty

Case 2



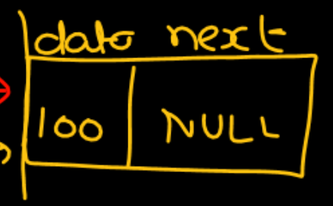
struct Node *Ptr; ⟹ it is a pointer to a node (not a node)

Ptr = START;

While(Ptr → next != NULL)

Ptr = Ptr → next;

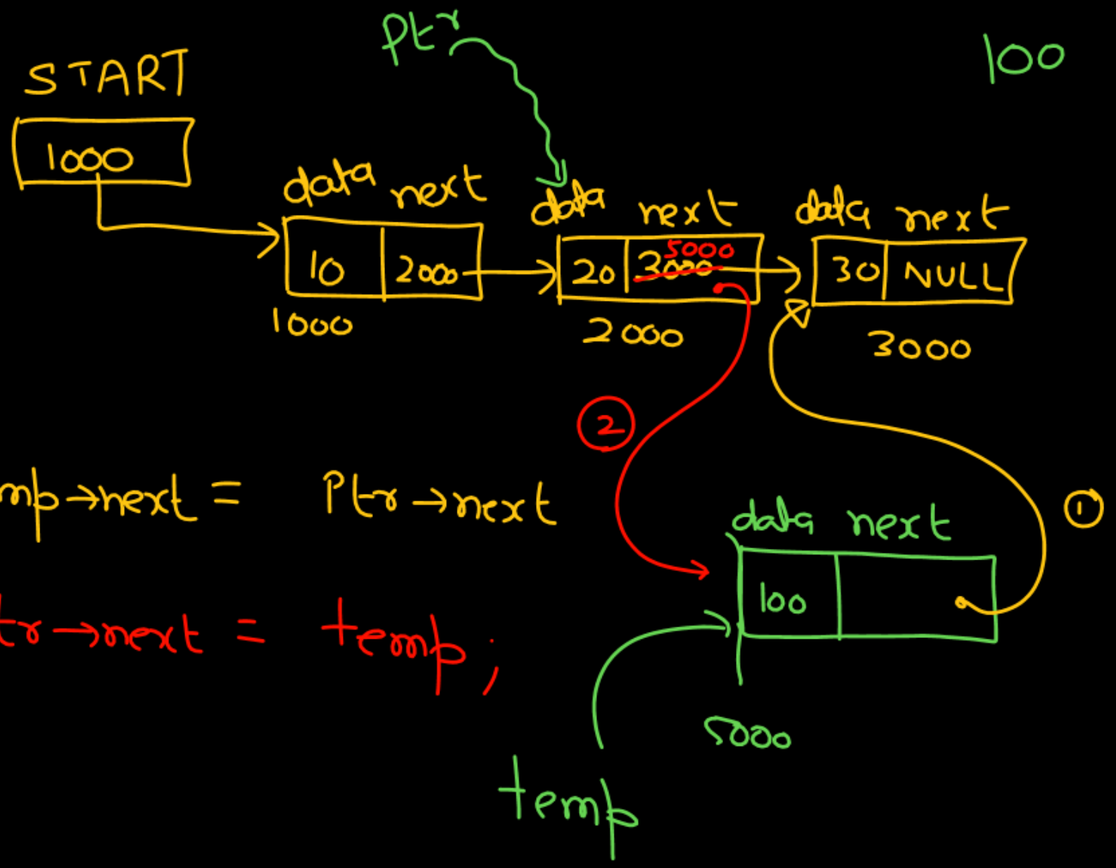
Ptr → next = temp;



temp

```
void Insert_at_end ( int key)
  {
struct Node *temp, *Ptr;
temp = malloc(sizeof(struct Node));
  if (temp) {
                temp -> data = key;
                temp -> next = NULL;
                if( START == NULL)
                    {
                    START = temp;
                    return;
                    }
                Ptr = START;
                While( Ptr -> next != NULL)
                    Ptr = Ptr -> next;
                Ptr -> next = temp;
                }
```

Q. Given a pointer to a node, insert an element after that node



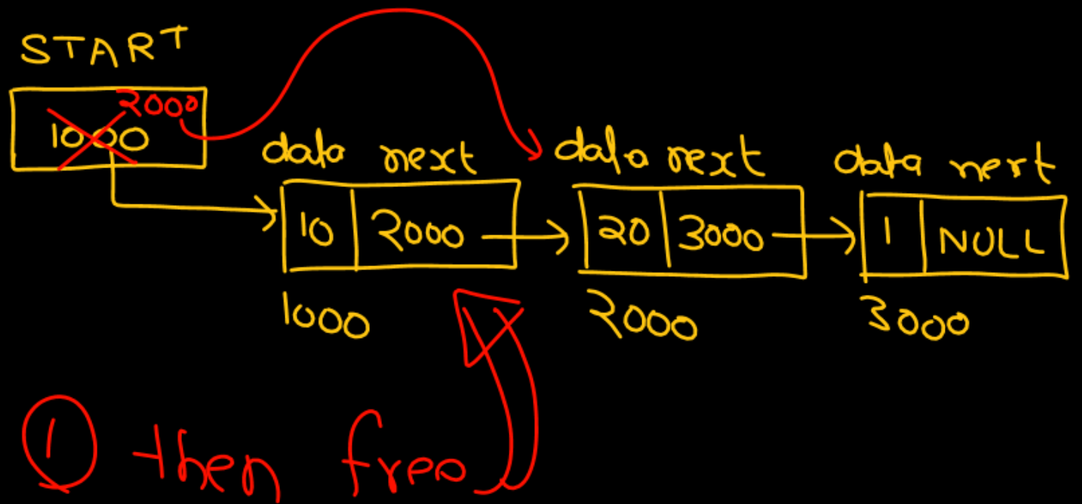
① temp->next = Ptr->next

Ptr->next = temp;

Deletion of an element (Node)

a) start

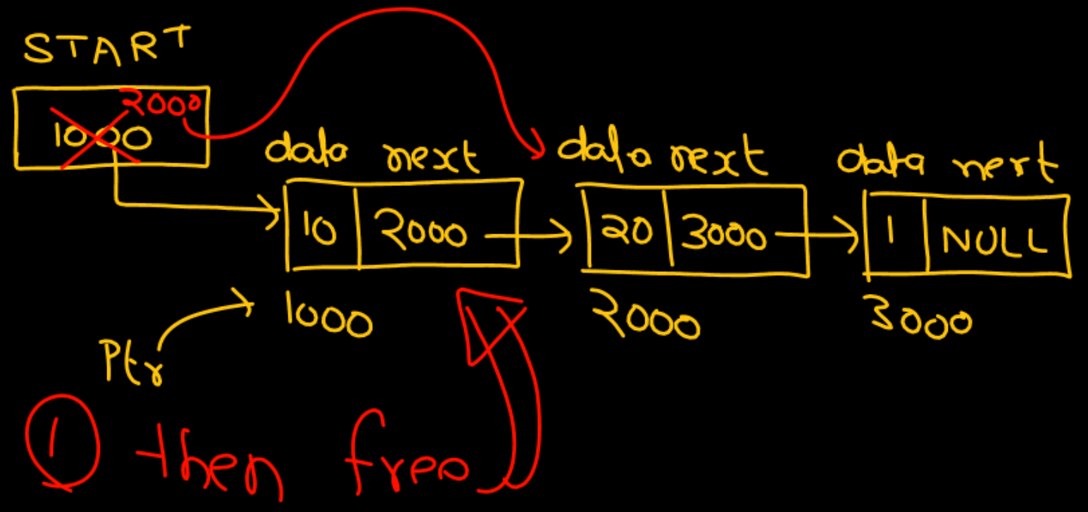
b) from end



After deletion of 1st Node START will contain address of 2nd node

When we know, L.L is not Empty

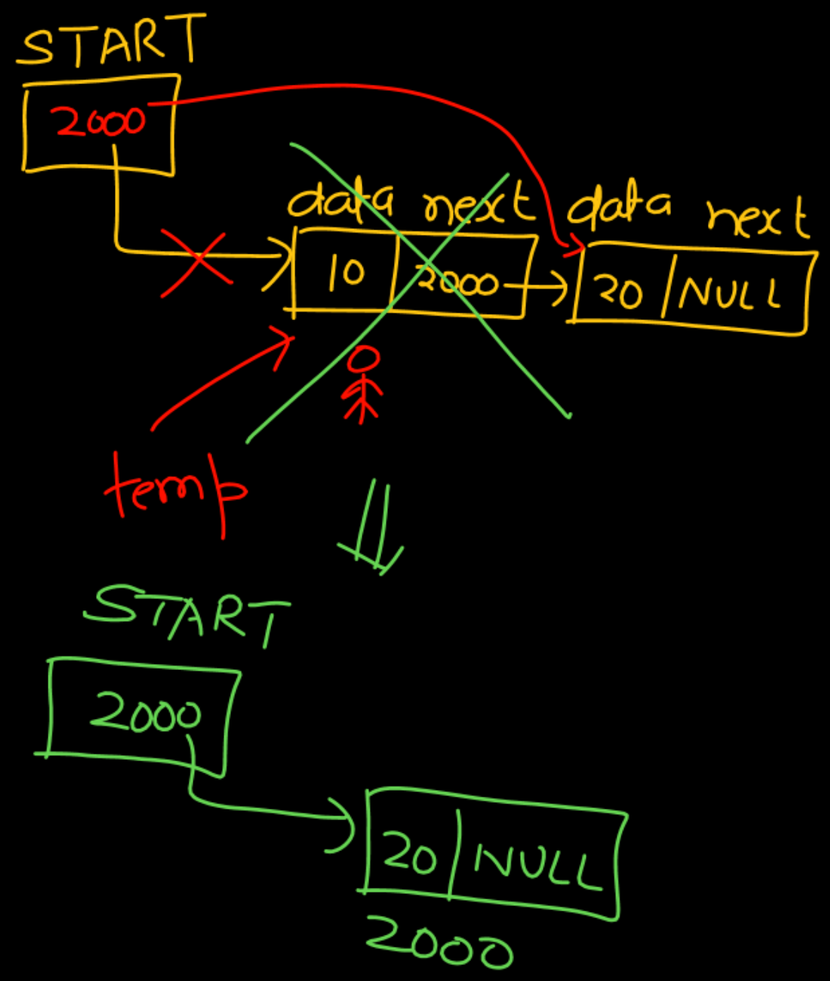
```
struct Node *Ptr;
Ptr = START;
START = START -> next;
free(Ptr)
```



After deletion of 1st Node START will contain address of 2nd node

Q

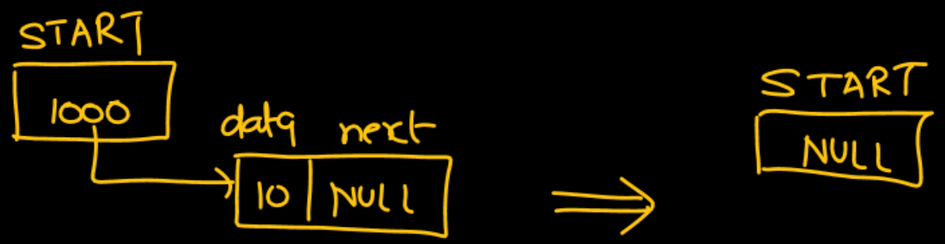
```
struct Node *temp ;
if (START==NULL)
    return ;

temp = START ;
START=START->next ;
free(temp);
```



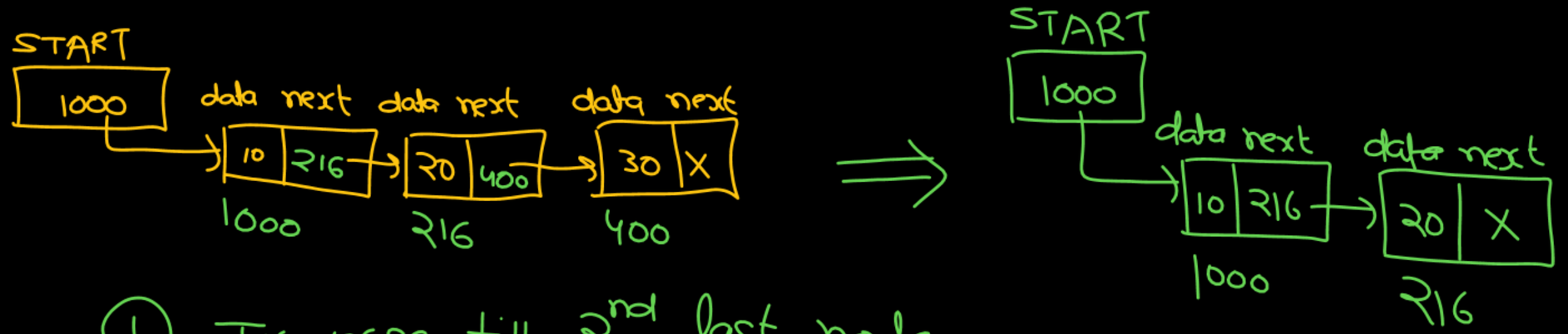
deletion from End

I.

II.



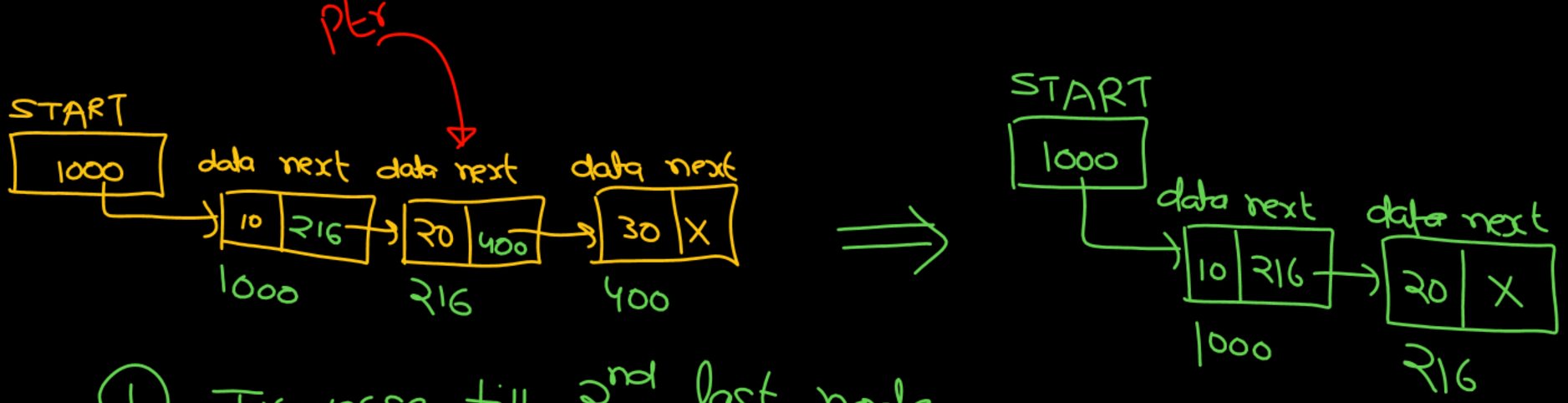
III



(1) Traverse till 2nd last node

Ptr ⇒ 2nd last node

III



① Traverse till 2nd last node

Ptr ⇒ 2nd last node

```
free(Ptr->next);
```

Ptr → next = NULL

1. count
2. Traversal
3. Insertion
4. Deletion
5. Search
6. last node ⇒ Print
7. first node ⇒ Print
8. reverse a linked list
9. Middle of a linked list
10. L.L. contains a loop
11. Intersection point of a Linked list.


THANK YOU GW
SOLDIERS !