CS & IT ENGINEERING

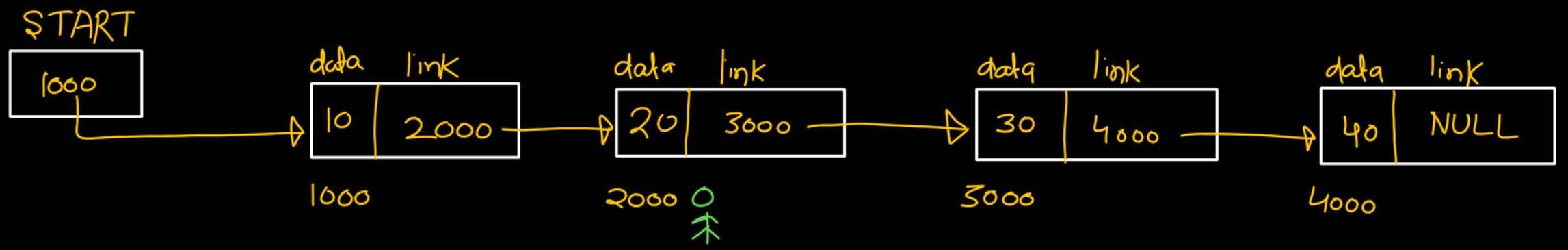
Data structure & Programming

Linked List

Lec- 02

By- Pankaj Sharma sir

Linked List -2



START : 1000 First Node address

* Pointer to a node (Pointer to a structure var. with 2 members)

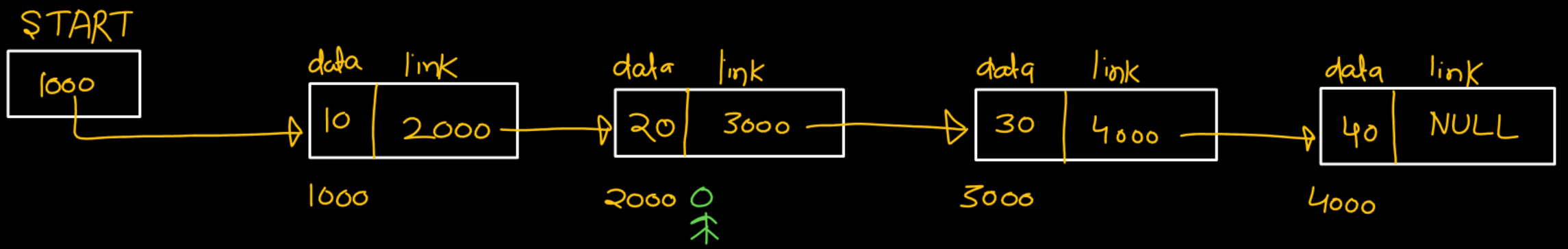
(i) START → data : 10

(ii) START → link : 2nd node address
Pointer to a 2nd node

✓

START → link → data : 2nd node data

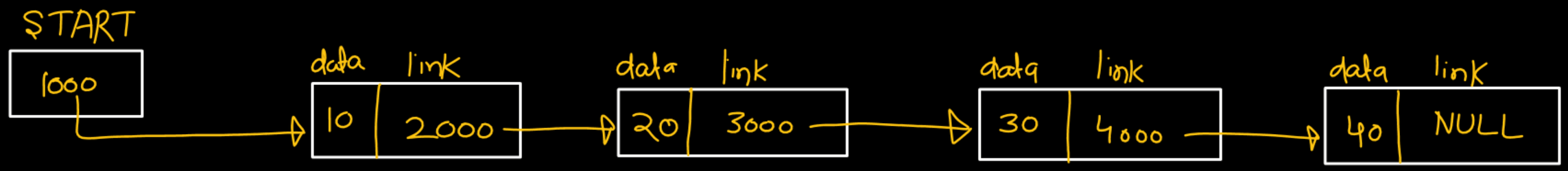
START → link → link : 3rd node address
Pointer to 3rd node

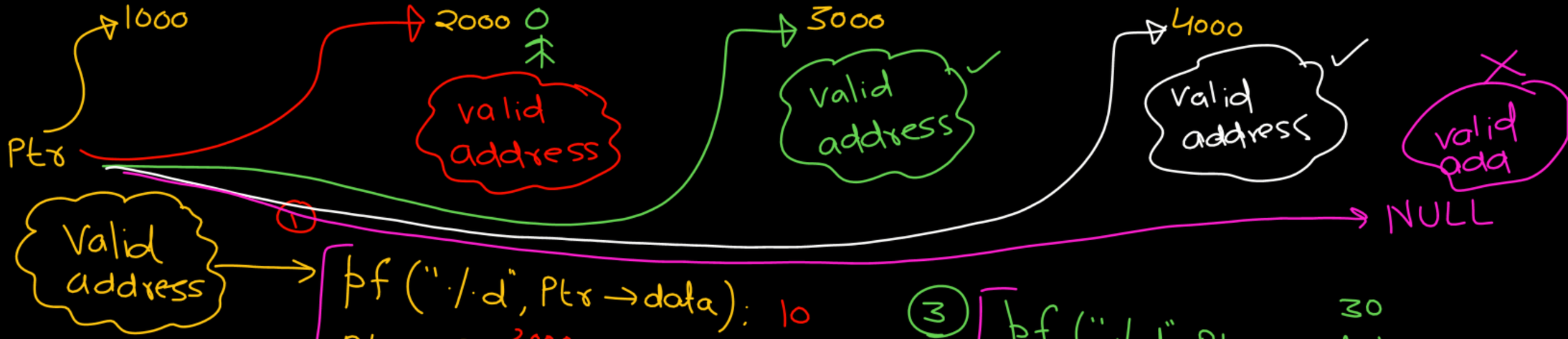


(i) START : Global

(ii) Given a linked list ⇒ START is given

① Given a linked list, **traverse** it.






```
pf("%d", Ptr->data);   10
Ptr = Ptr->link;       (2000)
```

```
Struct Node *Ptr;
Ptr = START;
```

② (3000)

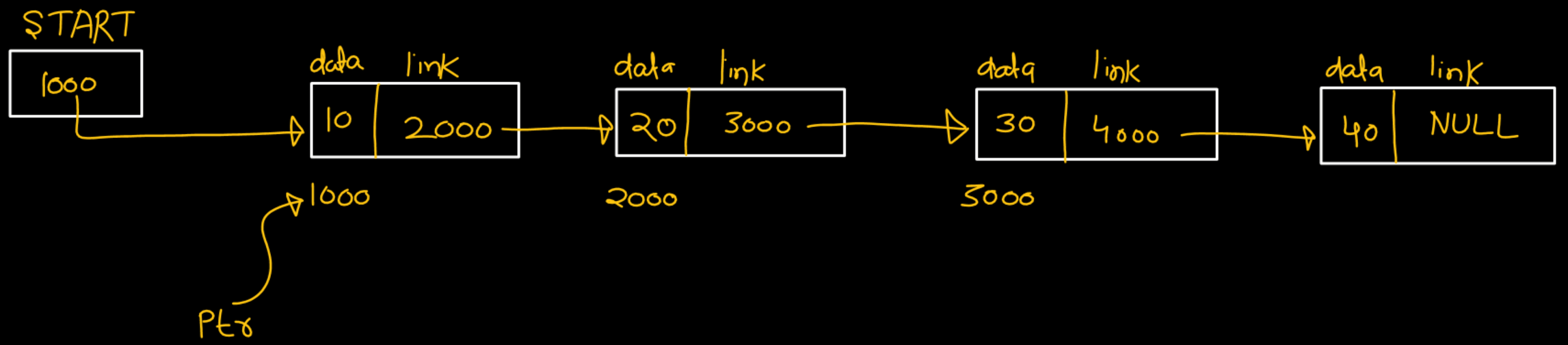
```
pf("%d", Ptr->data);   20
Ptr = Ptr->link;
```

③ (4000)

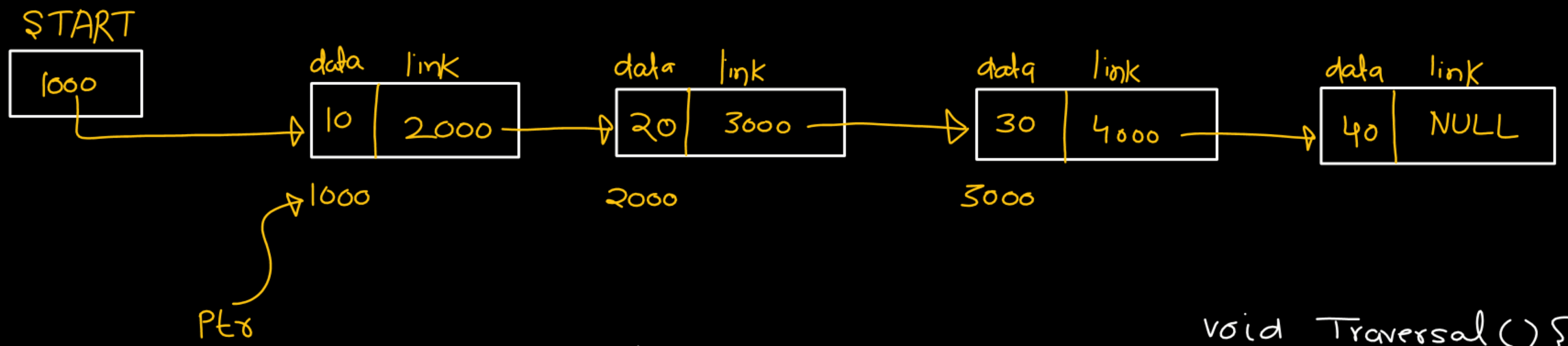
```
pf("%d", Ptr->data);   30
Ptr = Ptr->link;
```

④

```
pf("%d", Ptr->data);   40
Ptr = Ptr->link;
```



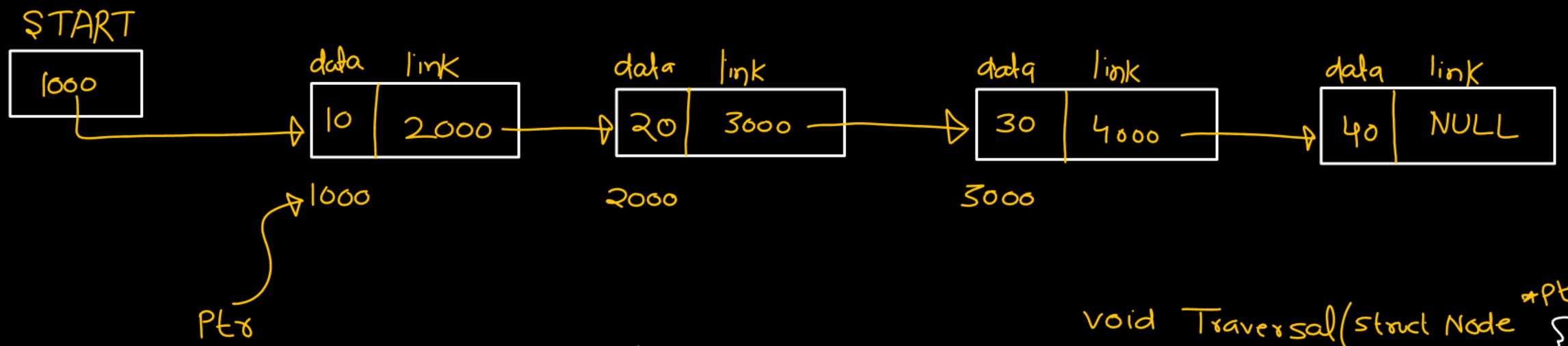

```
struct Node *Ptr ;
Ptr = START ;

While( Ptr != NULL)
     {
        pf (" %d", Ptr -> data);
        Ptr = Ptr -> link;

     }
```




```
struct Node{
        int data;
    struct Node *link;
        }*START=NULL;
```

```
void main(){
    ==
Traversal();
    ==
    ==
}
```

```
void Traversal(){
    struct Node *Ptr;
    Ptr = START;
    While(Ptr != NULL)
        {
    >> pf("%d", Ptr->data);
    Ptr = Ptr->link;
        }
}
```

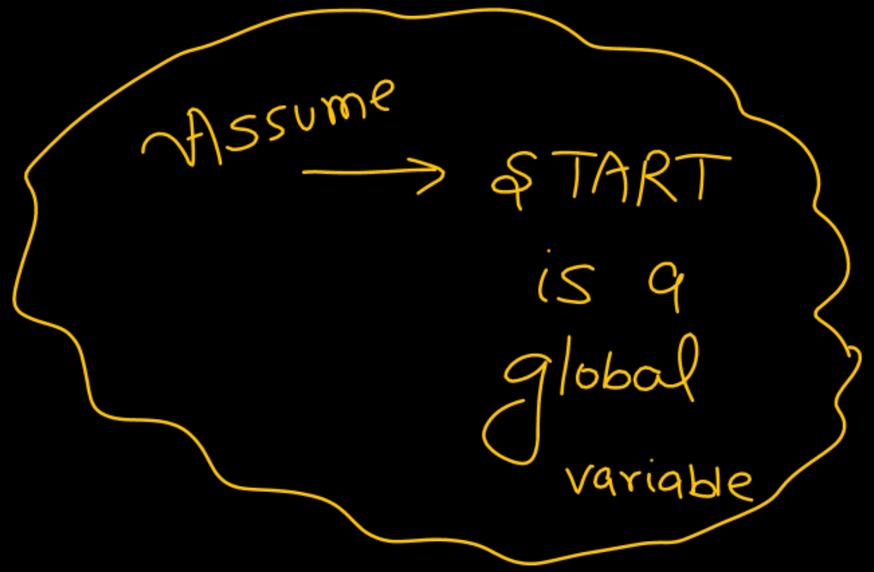


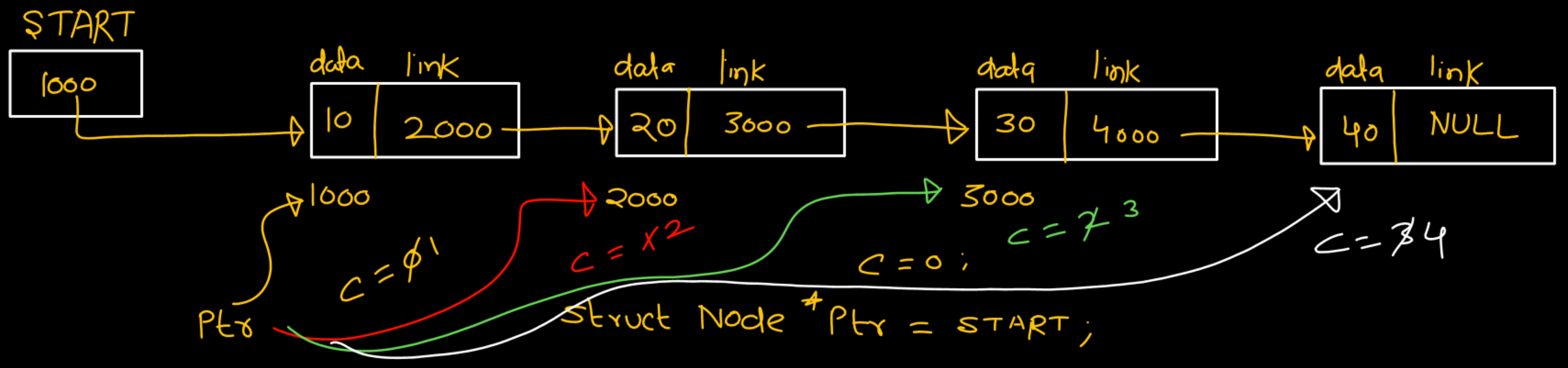

```
struct Node{
        int data;
    struct Node *link;
        }
```

```
void main(){
    struct Node *START=NULL;
    {
    }
Traversal(START);
}
```

local var.

```
void Traversal(struct Node *Ptr)
{
while(Ptr!=NULL)
    {
pf("%d",Ptr->data);
Ptr=Ptr->link;
    }
}
```


Assume
START
is a
global
variable



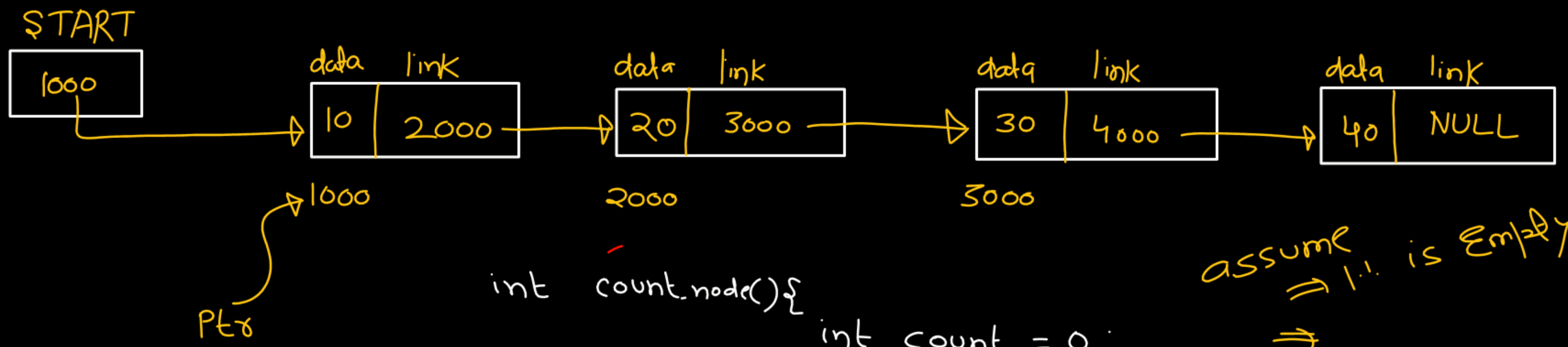
2 Given a l.l., count the no. of nodes in given l.l.

① Ptr ⟹ valid add ✓

c++ ;

Ptr = Ptr → link

②



2 Given a l.l., count the no. of nodes in given l.l.

```
int count_node(){
    int count = 0;
    struct Node *Ptr;
    Ptr = START;
    While(Ptr != NULL)
    {
        count++;
        Ptr = Ptr->link;
    }
    return count;
}
```

assume ⇒ l.l. is Empty

⇒ Ptr = NULL

NULL → link

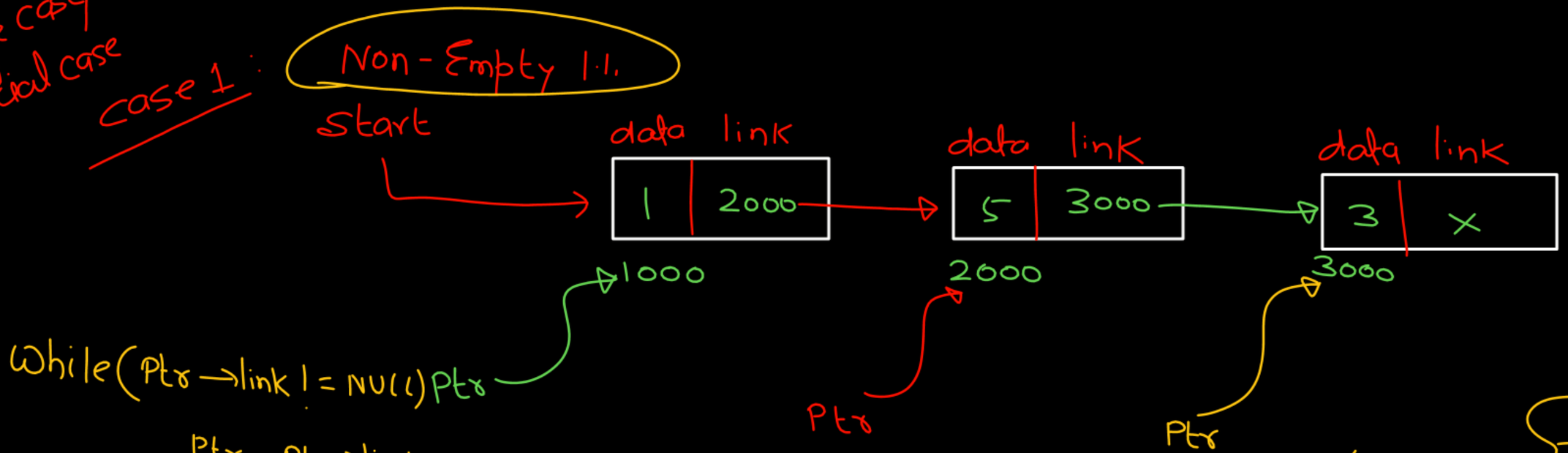
ishan

```
While(Ptr->link != NULL)
{
```

3: Given a linked list, print the data of last node.

edge case/
special case

case 1 : Non-Empty L.L.




```
While(Ptr->link != NULL) Ptr = Ptr->link;
print Ptr->data
```

(i) Ptr->link != NULL ✓ true
Ptr = Ptr->link

(ii) Ptr->link != NULL
Ptr = Ptr->link

(iii) Ptr->link != NULL false

चाहता है

last node data

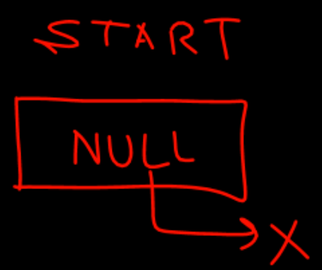
When last node exist

```c
struct Node *Ptr ;
Ptr = START ;
While( Ptr → link != NULL)
        Ptr = Ptr → link;
printf("%d", Ptr → data);
```

what if l.l. is Empty

1) Ensure that last node exist

```
void printlast(){
    struct Node *Ptr;
    Ptr = START;
    if(START==NULL)          // No node
        return;              // No last node
    while(Ptr->link!=NULL)
        Ptr = Ptr->link;
    printf("%d", Ptr->data);
}
```



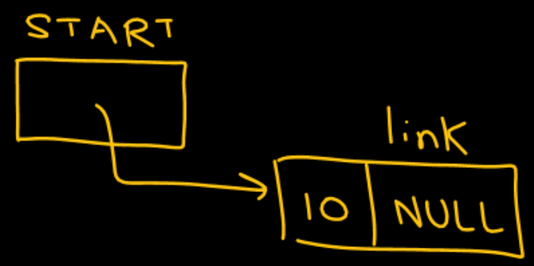
4. Given a linked list, print second last node data.

Case 1:

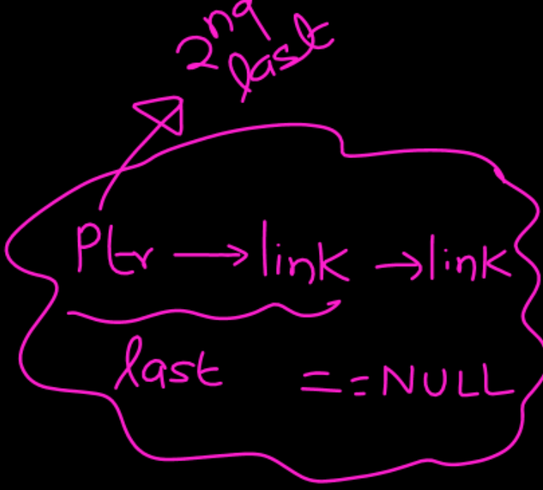
No second last node } ⇒ do nothing

```
if(START == NULL ||
   START → link == NULL)
    return ;
```

Case 2:



No second last node



Ptr → link → link != NULL

Ptr ⇒ update

Ptr

ptr

0 node

Ptr → link → link != NULL false

1 node

if (START == NULL || START → link == NULL)

return

Ensured that atleast 2 node exist ⟹

True

if (START == NULL) if (0)

Bhelpoori

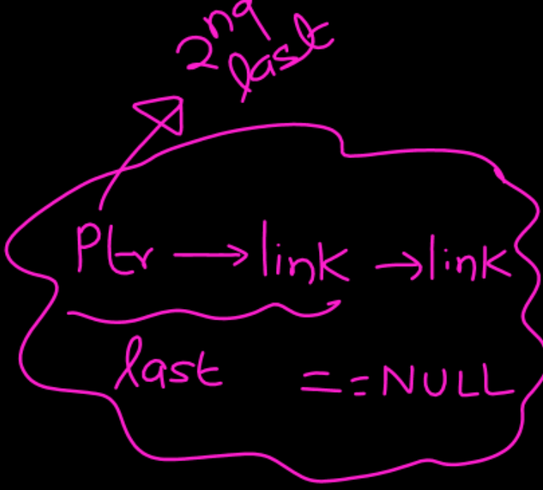
मत बनाओ

Vidyut

NULL → link ⇒ Error

&& START → link == NULL)

Evaluate



✓ Ptr → link → link != NULL

Ptr

ptr

0 node

Ptr → link → link != NULL false

1 node

Ptr ⇒ update

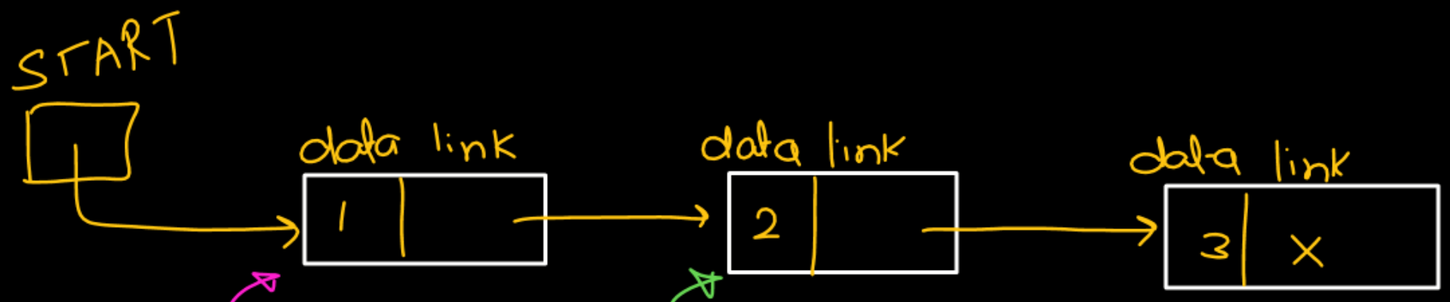
```
if (START == NULL || START → link == NULL)
    return
```

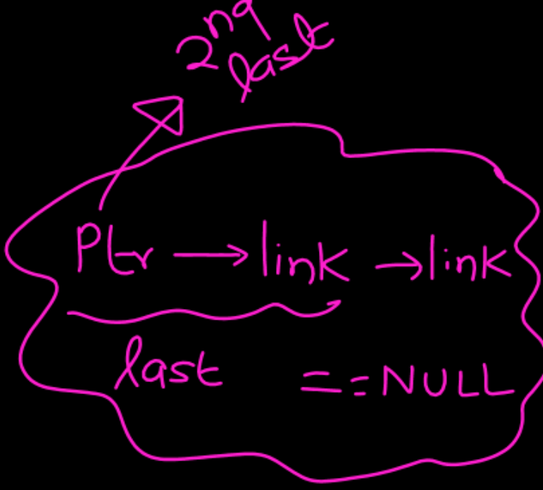
Ensured that atleast 2 node exist ⟹

linked list is Empty → Tiger shroff

NULL → link

```
if ( START → link == NULL || START == NULL)
    return
```





Ptr → link → link != NULL

Ptr ⇒ update

Ptr

ptr

0 node

Ptr → link → link != NULL false

1 node

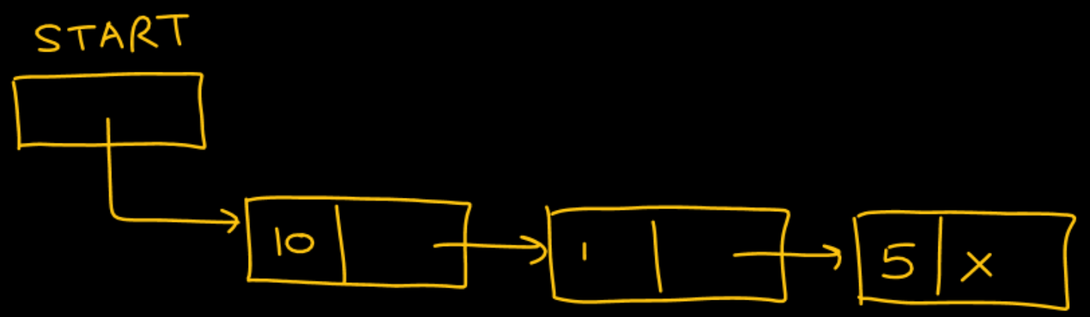
```
if (START == NULL || START → link == NULL)
    return
```

Ensured that atleast 2 node exist ⟹

```
Ptr = START;
While ( Ptr → link → link != NULL)
    Ptr = Ptr → link;
bf("%d", Ptr → data);
```

5: Given a linked list and a key. find whether the key is present in the linked list or not.

i/p:



key : 102

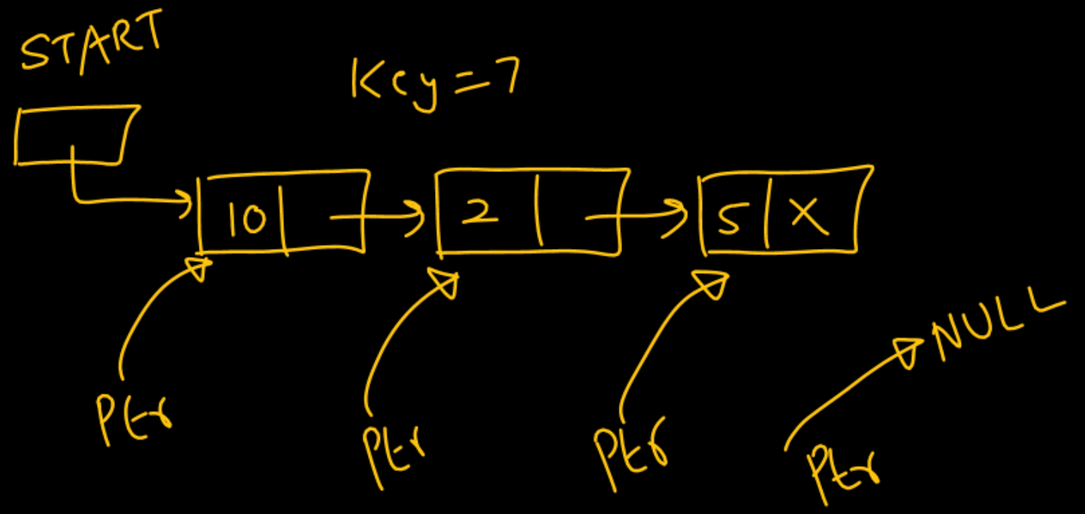
o/p : NO

i/p

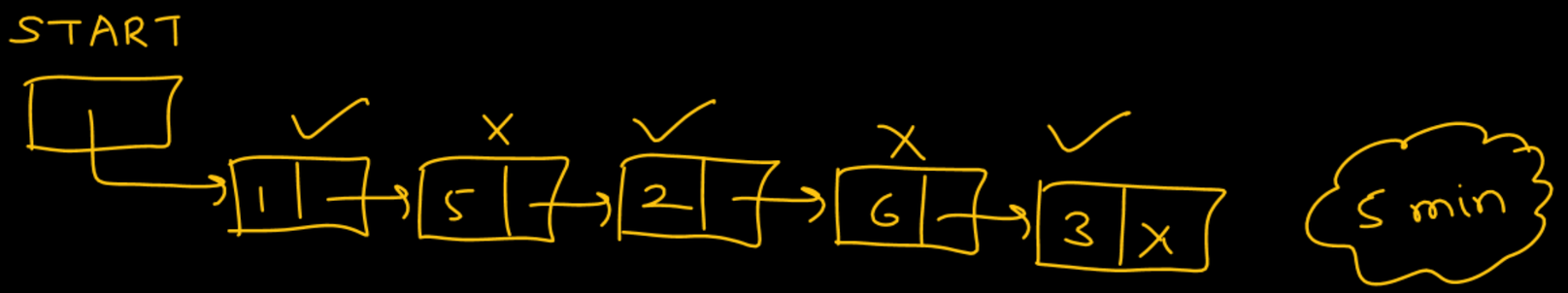
key : 7

o/p : Yes

```
void Search(int key){
    struct Node *Ptr = START;
    while(Ptr != NULL)
    {
     if(Ptr->data == key)
                {
                printf("Yes");
                return;
                }
          Ptr = Ptr->link;
    }
bf("No");
}
```



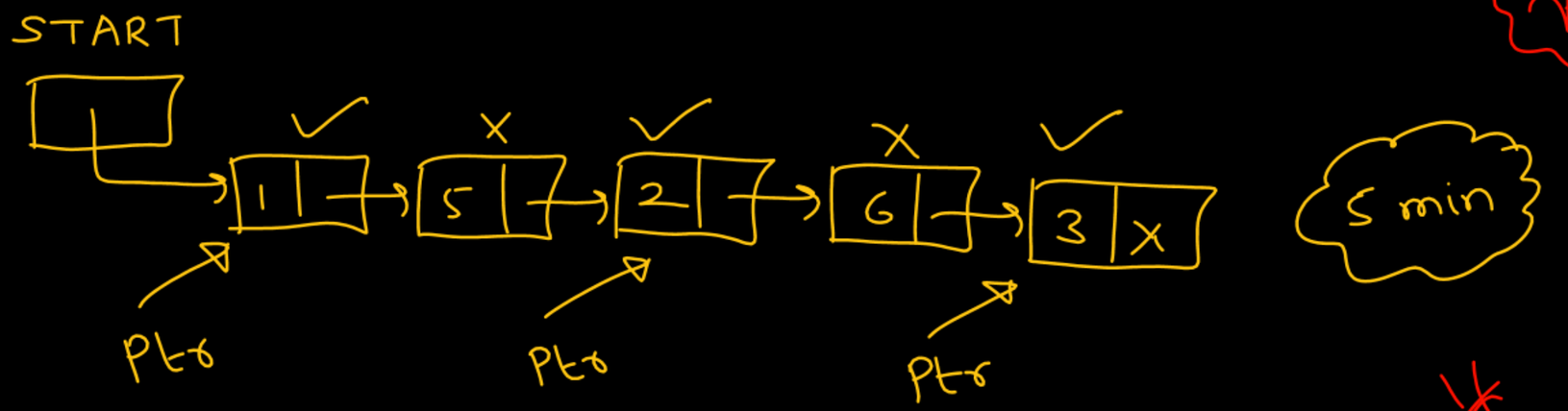
6. Given a l.l, write a code to print alternate node data.



O/P : 1 2 3

O/P : 1 2

6. Given a l.l, write a code to print alternate node data.



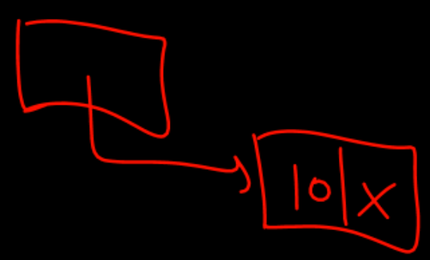
Analyse

5 min

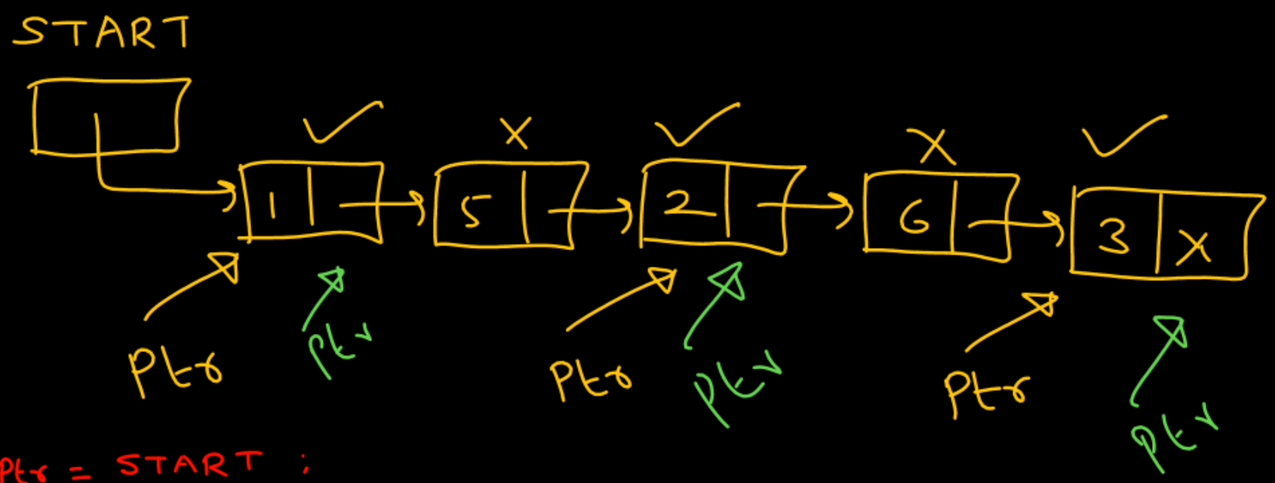
```
struct Node *Ptr = START ;

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

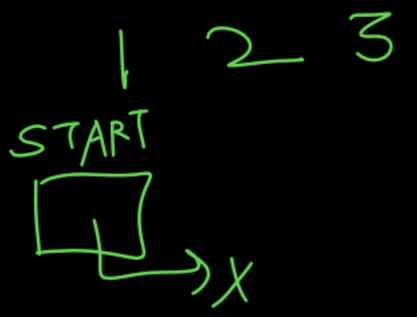
dict* {}

Empty l.l

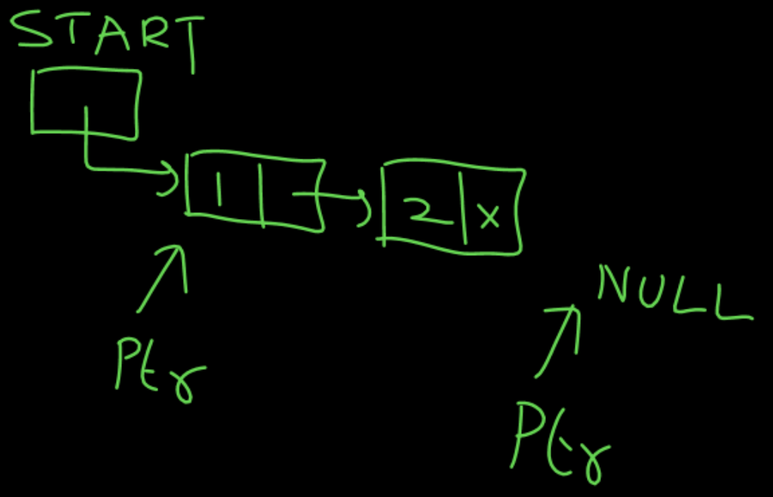


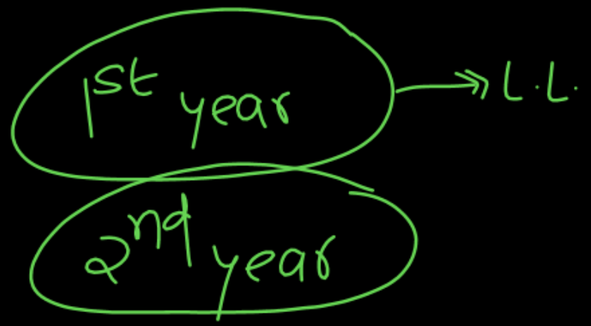
6. Given a l.l, write a code to print alternate node data.






```
struct Node *Ptr = START ;

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

if (Ptr == NULL)
        return;
if (Ptr->link == NULL)
    pf("%d", Ptr->data);
```





Easy Problem

2 session

level up

live - Code

08:30 PM

2 min wait

⇒


THANK YOU GW
SOLDIERS !