CS & IT ENGINEERING

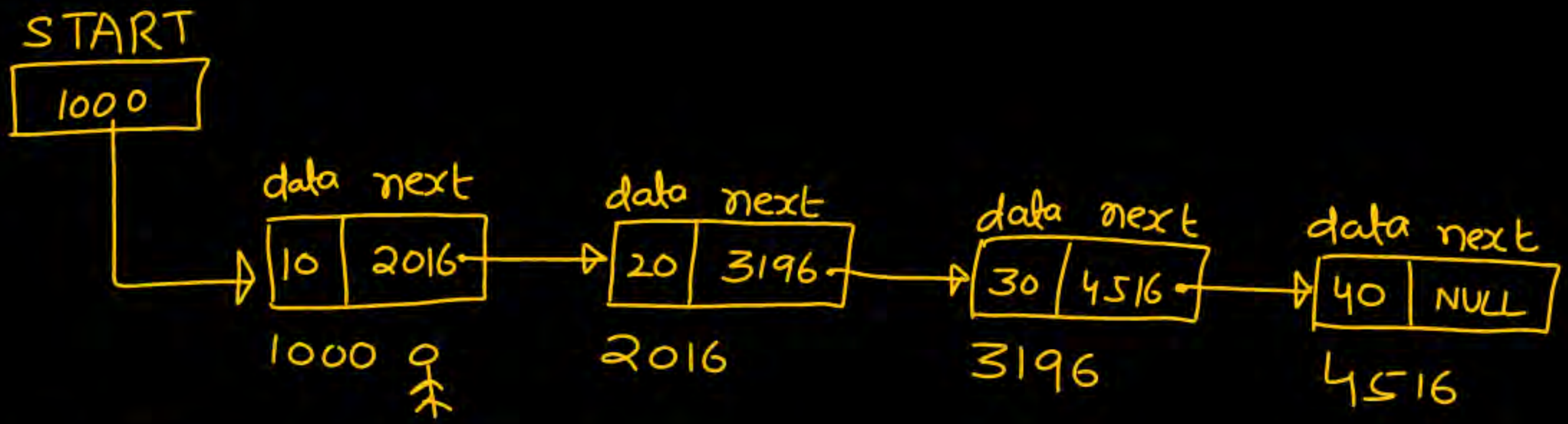
Data Structure

Linked List
Chapter- 3
Lec- 02

By- Pankaj Sharma sir

TOPICS TO BE COVERED

- Linked List-II



(i) START : Pointer to first node

Pointer to a structure type variable (Member1 data , Member2 next)

printf("%d", START → data); first node data ⇒ 10

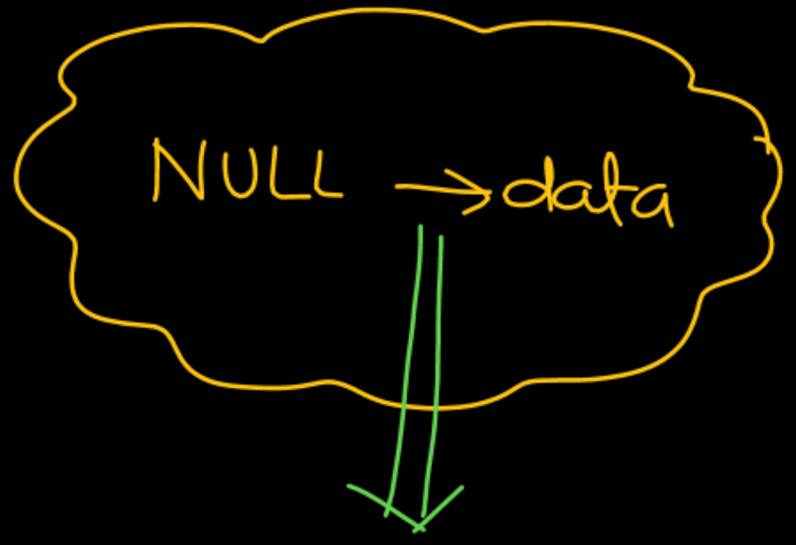
(ii) START → next : Memory for 2016

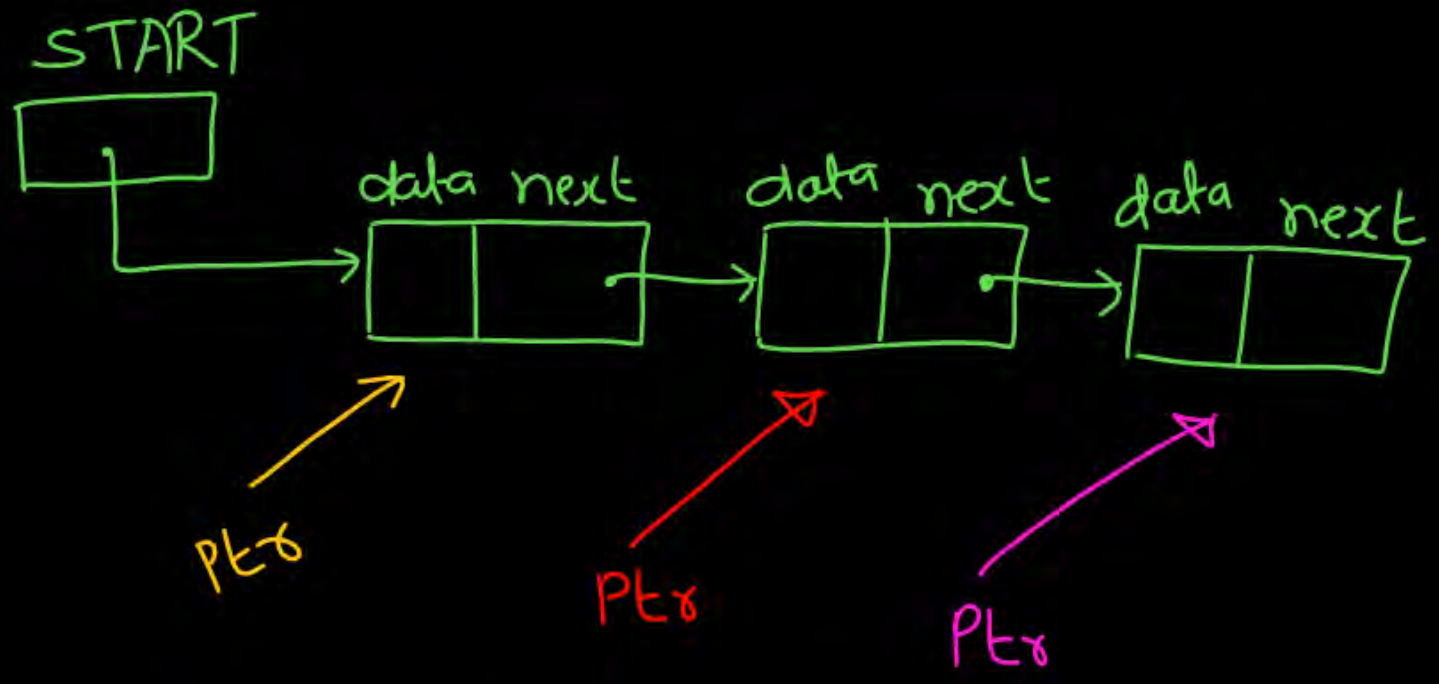
: Pointer to second node

(START → next) → data : 20

(START → next) → next : Pointer to 3rd node l.l. is Empty

```
[ Ptr = START;

      Ptr -> data;
]
```





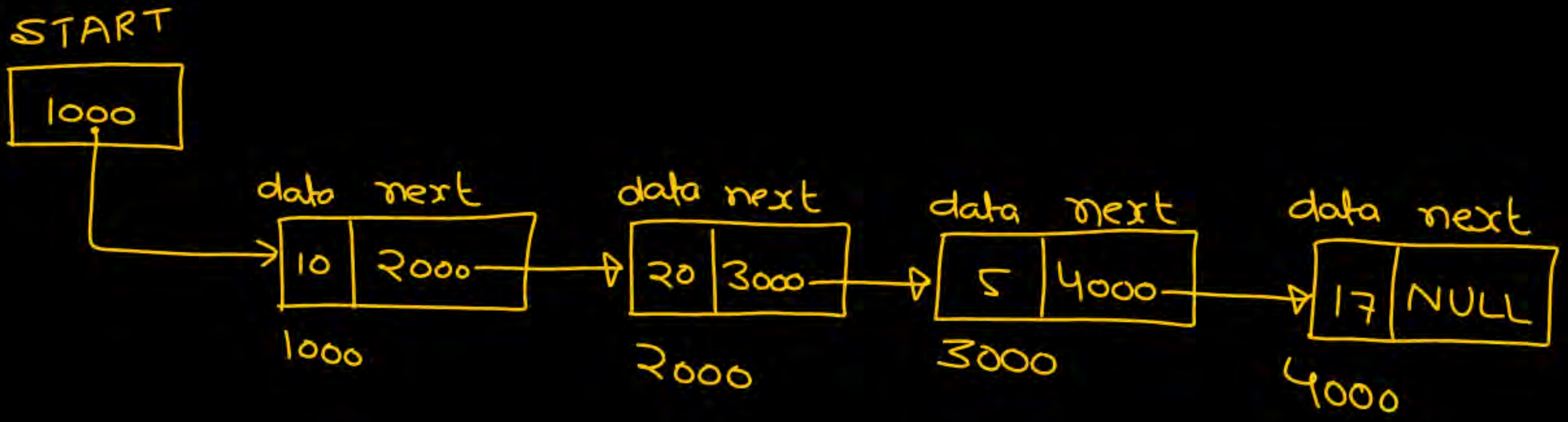
Ptr → next : Valid address

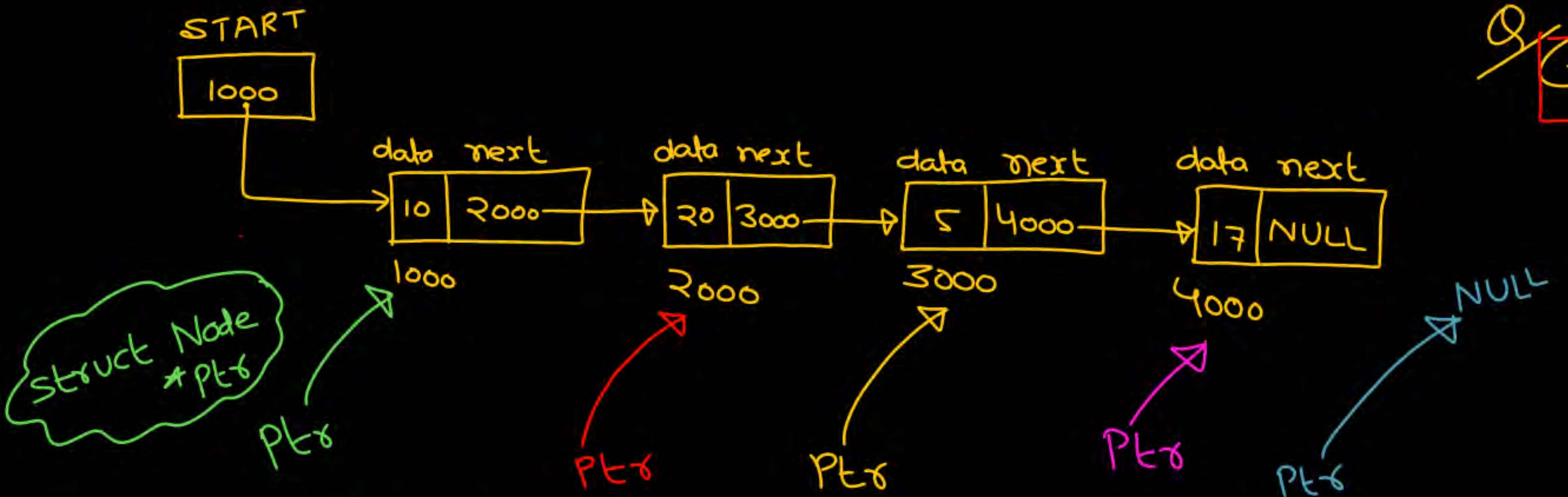
Ptr → next : valid address

$$\boxed{\text{Ptr} \rightarrow \text{next} : \text{NULL}}$$

START : global variable

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





Q. Given a linked list, traverse the l.l.

operation

v) Ptr : Valid ✗ → NULL STOP

(i) Ptr : valid address

```
[ printf("%d", Ptr→data);
  Ptr = Ptr→next;
```

(ii) Ptr : valid address

```
[ printf("%d", Ptr→data);
  Ptr = Ptr→next;
```

(iii) Ptr : Valid address

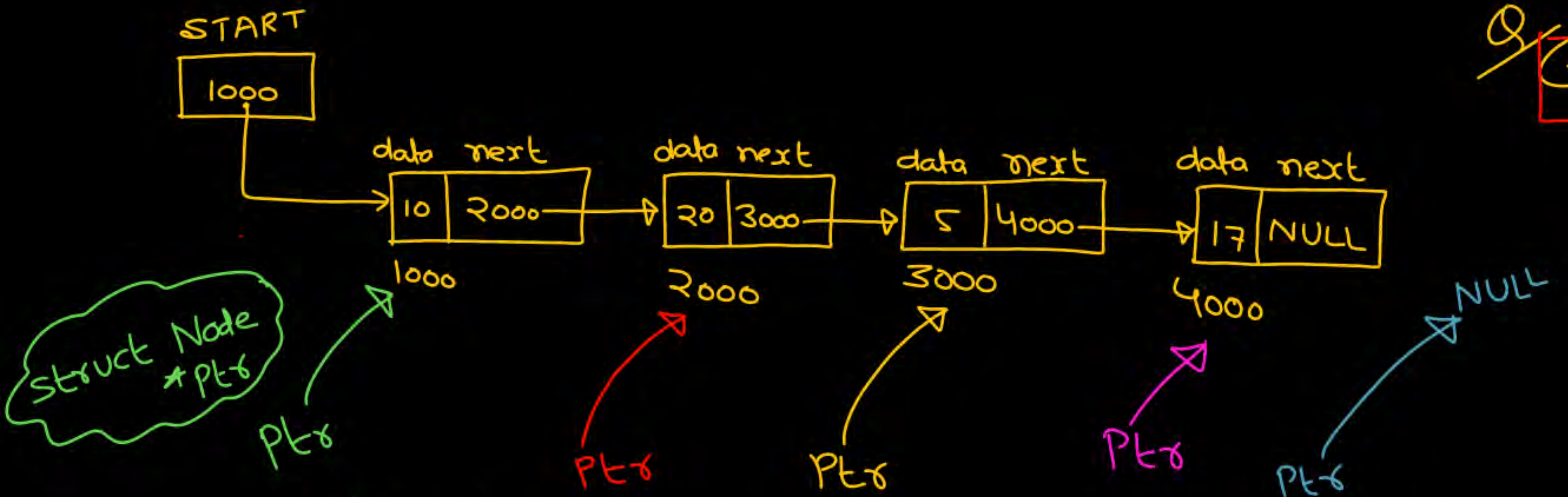
```
[ printf("%d", Ptr→data);
  Ptr = Ptr→next;
```

(iv) Ptr : Valid address

```
[ printf("%d", Ptr→data);
  Ptr = Ptr→next;
        NULL;
```

Q. Given a linked list, traverse the l.l.

operation



v) Ptr : Valid ✗
→ NULL
STOP

(i) Ptr : valid address

```
[ printf("%d", Ptr → data);
  Ptr = Ptr → next;
```

(ii) Ptr : valid address

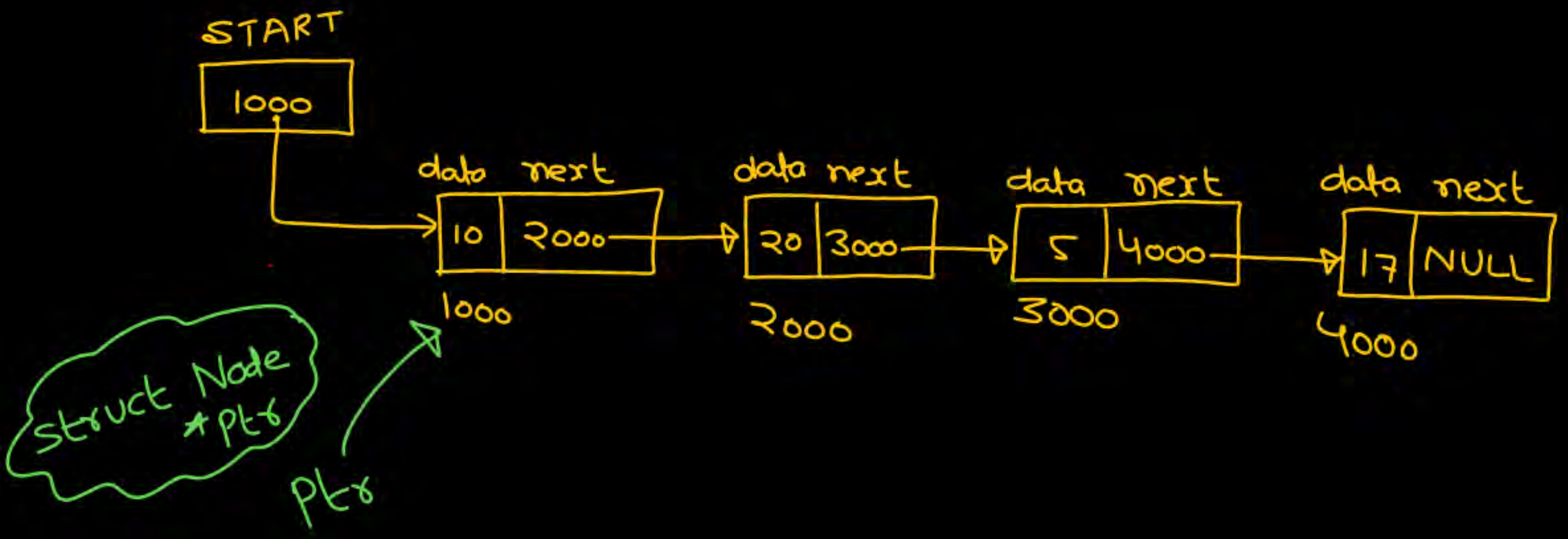
```
[ printf("%d", Ptr → data);
  Ptr = Ptr → next;
```

(iii) Ptr : Valid address

```
[ printf("%d", Ptr → data);
  Ptr = Ptr → next;
```

(iv) Ptr : Valid address

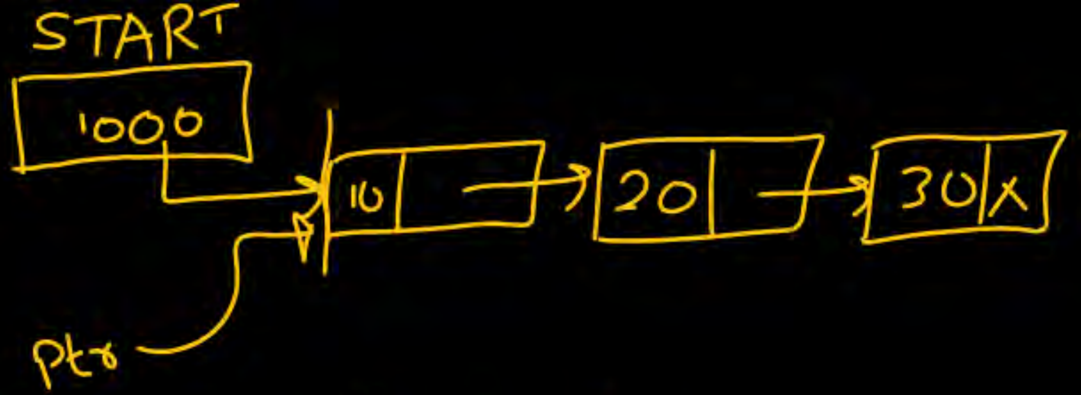
```
[ printf("%d", Ptr → data);
  Ptr = Ptr → next;
        NULL;
```



```
While (ptr != NULL)
{
    printf("%d", ptr->data);
    ptr = ptr->next;
}
```

```
struct Node{
        ____
        ____

        }*START=NULL;

void  Traversal( ){

    struct Node *Ptr;
    Ptr = START;
        While(ptr!=NULL){
          printf("%d", Ptr->data);
          Ptr = Ptr->next;
                 }

                  }
```

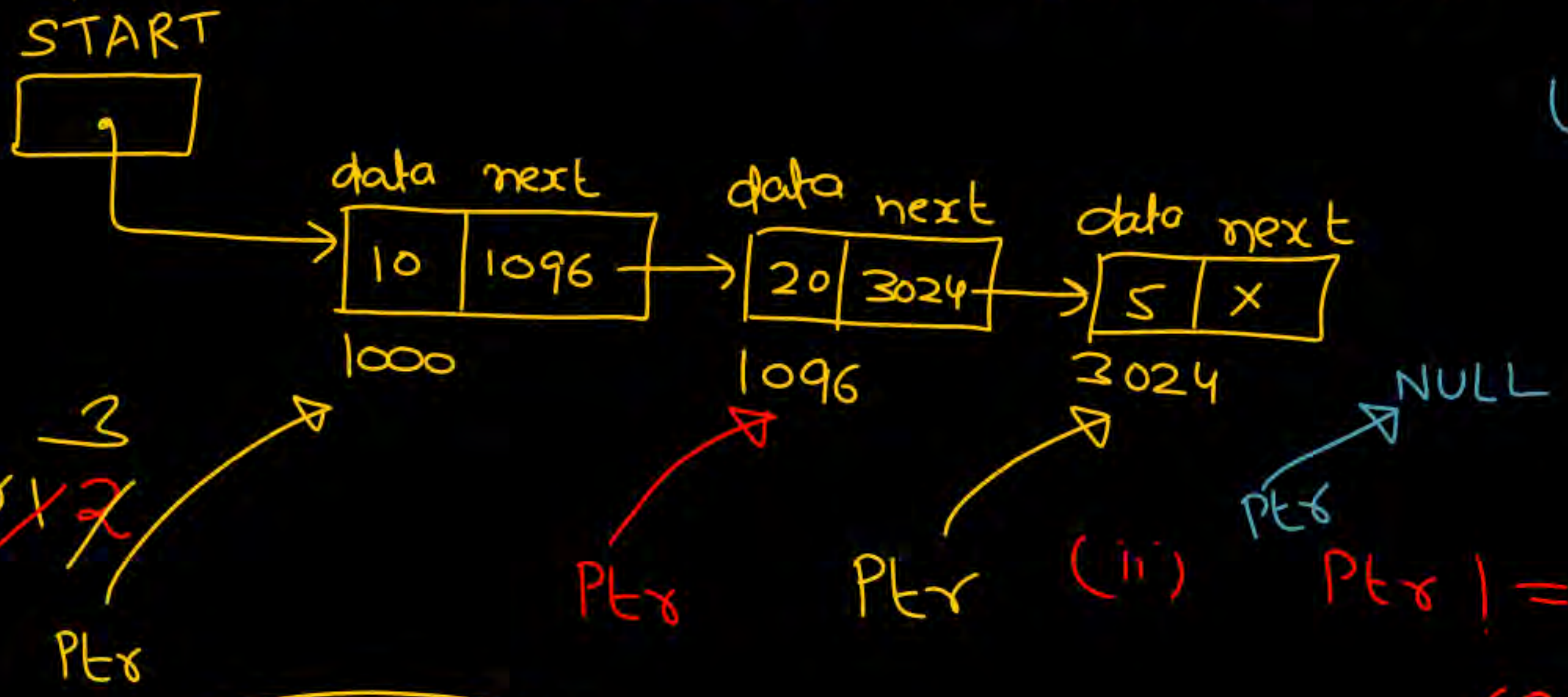
```
void  main( ){

        ____
    Traversal( );
        ____
        ____

        }
```



```
void    Traversal(struct Node *ptr)
  {

      While(ptr != NULL)
        {
        printf("%d", ptr->data);
        ptr = ptr->next;
        }

  }
```

```
void main(){

  struct Node *START = NULL;

  Traversal( START )
              1000

}
```

2. Given a linked list, count the no. of nodes in L.L.



count = ~~0~~ ~~1~~ ~~2~~ 3

(i) Ptr != NULL → True

count++;
Ptr = Ptr → next;

(ii) Ptr != NULL ✓

count++;
Ptr = Ptr → next;

(iii) Ptr != NULL ✓

count++;
Ptr = Ptr → next;

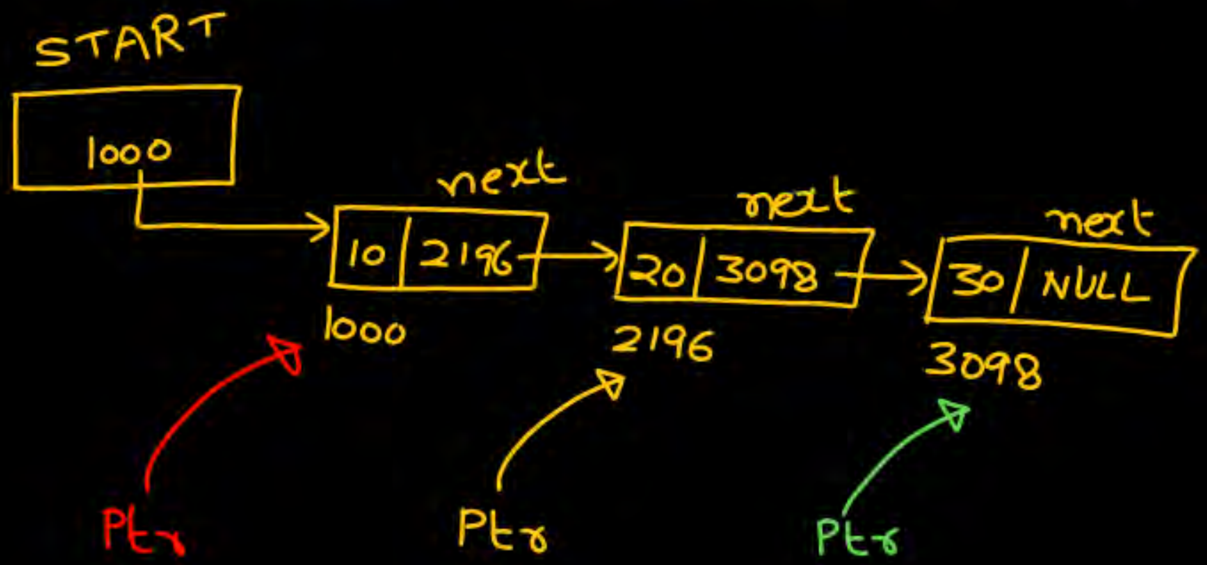
(iv) Ptr != NULL

$\boxed{\text{false}}$

```
int
~~void~~ Counting(){
        int count =0;
        struct Node *Ptr;

        Ptr = START;

        While(Ptr != NULL)
            {
                count++;
                Ptr = Ptr ->next;
            }
         return count;

        }
```

Q3 Given a linked list, print the data of last node in L.L (5 min).

गलत code है



(i) Ptr → next != NULL

Ptr = Ptr → next;

(ii) Ptr → next != NULL

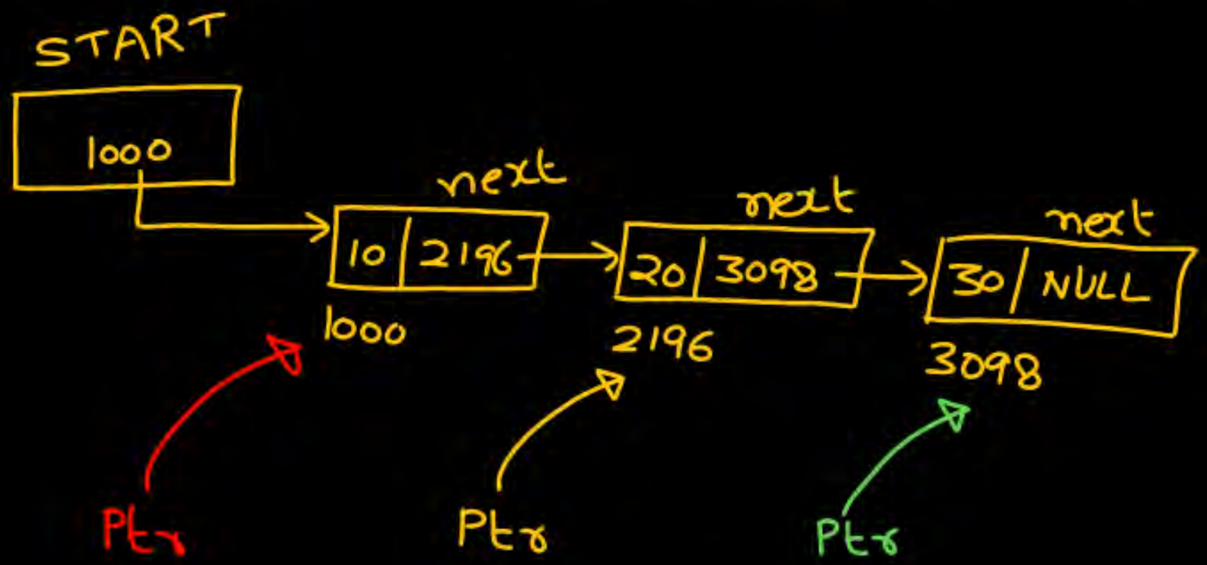
Ptr = Ptr → next;

(iii) Ptr → next != NULL {false}

```
struct Node* Ptr;
    Ptr = START;
    while (Ptr → next != NULL)
        Ptr = Ptr → next;
    printf("%d", Ptr → data);
```

What if L.L is Empty

Given a linked list, print the data of last node in L.L (5 min).



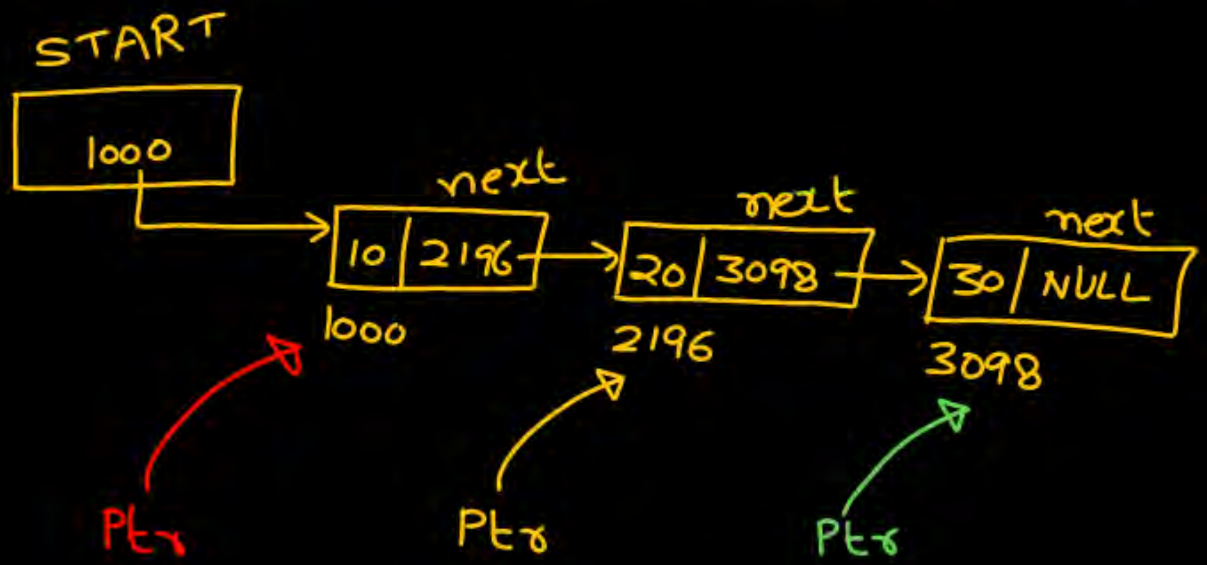
```
Struct Node* Ptr;
    Ptr = START;
While (Ptr->next != NULL)
        Ptr = Ptr->next;
printf("%d", Ptr->data);
```

NULL→next

Ud ke laat Marega

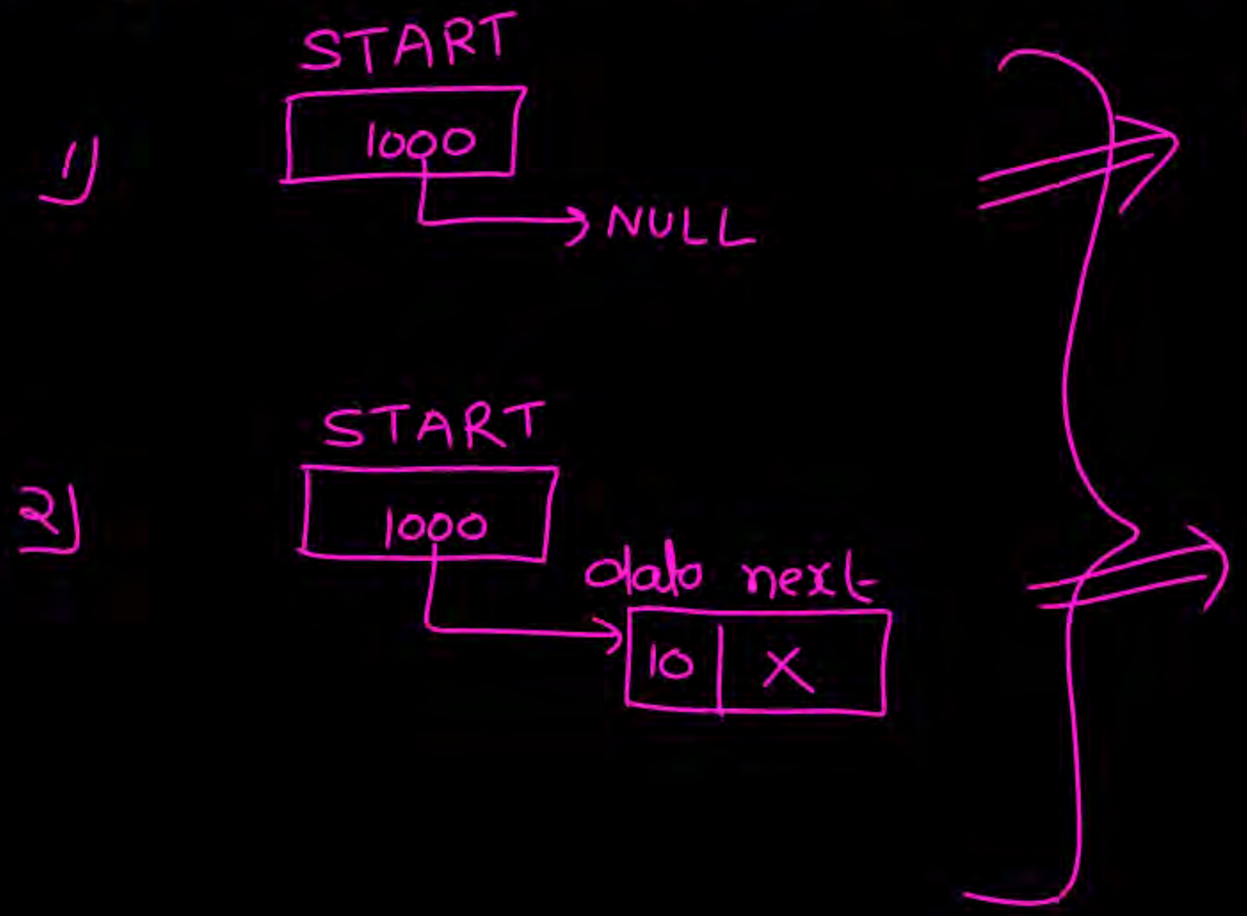
What if L.L is Empty

Given a linked list, print the data of last node in L.L (5 min).



```
Ptr = START;
if ( START == NULL)
        return;
→       while (Ptr→next != NULL)
                Ptr = Ptr→next;
        printf("%d", Ptr→data);
```

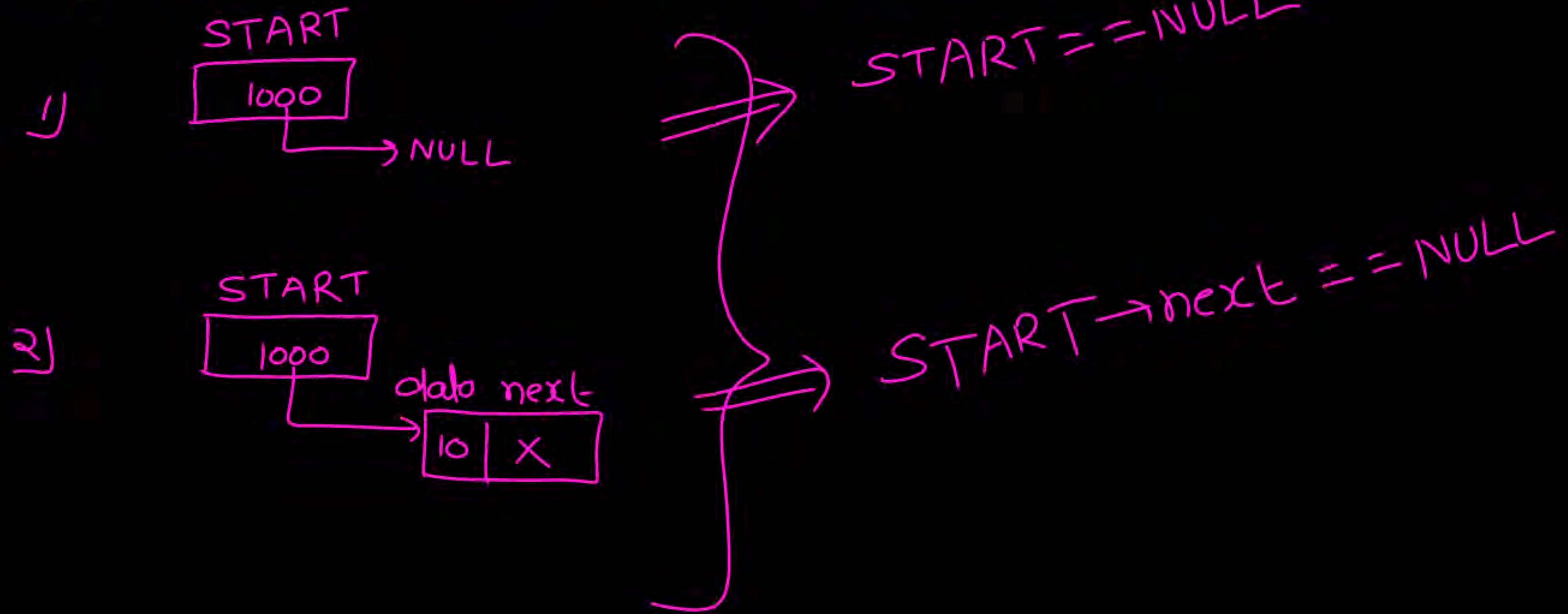
4 Given a linked list, print data in second last node.



START == NULL

START → next == NULL

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

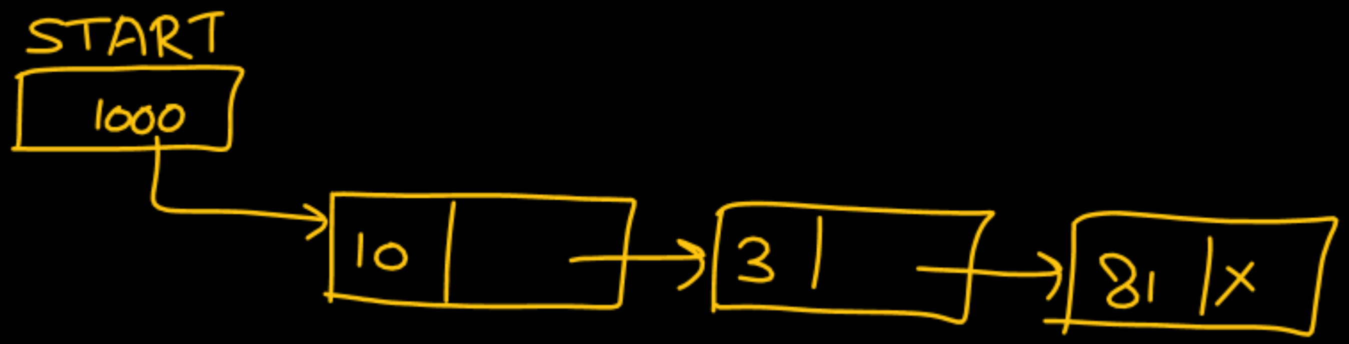


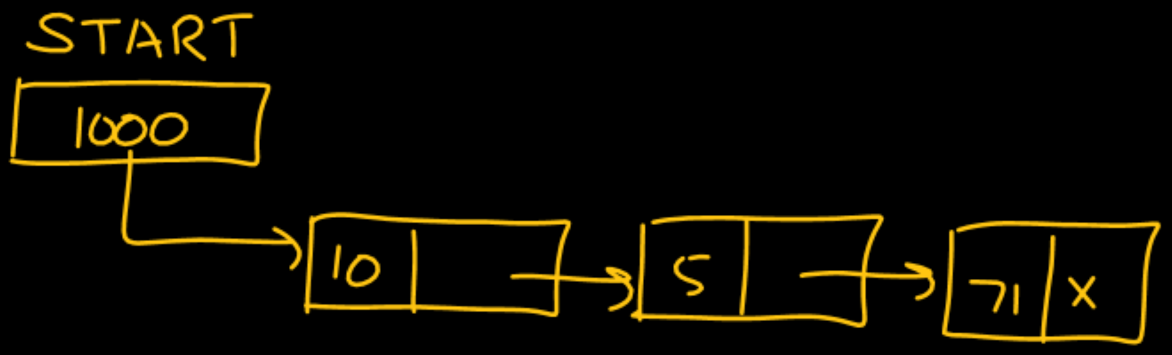
```
if( START==NULL || START→next==NULL)
    return;
```

Ensured that atleast 2 nodes are present ⟹

```
      0 Nodes                    1 Node
if(START = = NULL || START→next = = NULL)
                 return;

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

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





Key : 117

O/P : NO

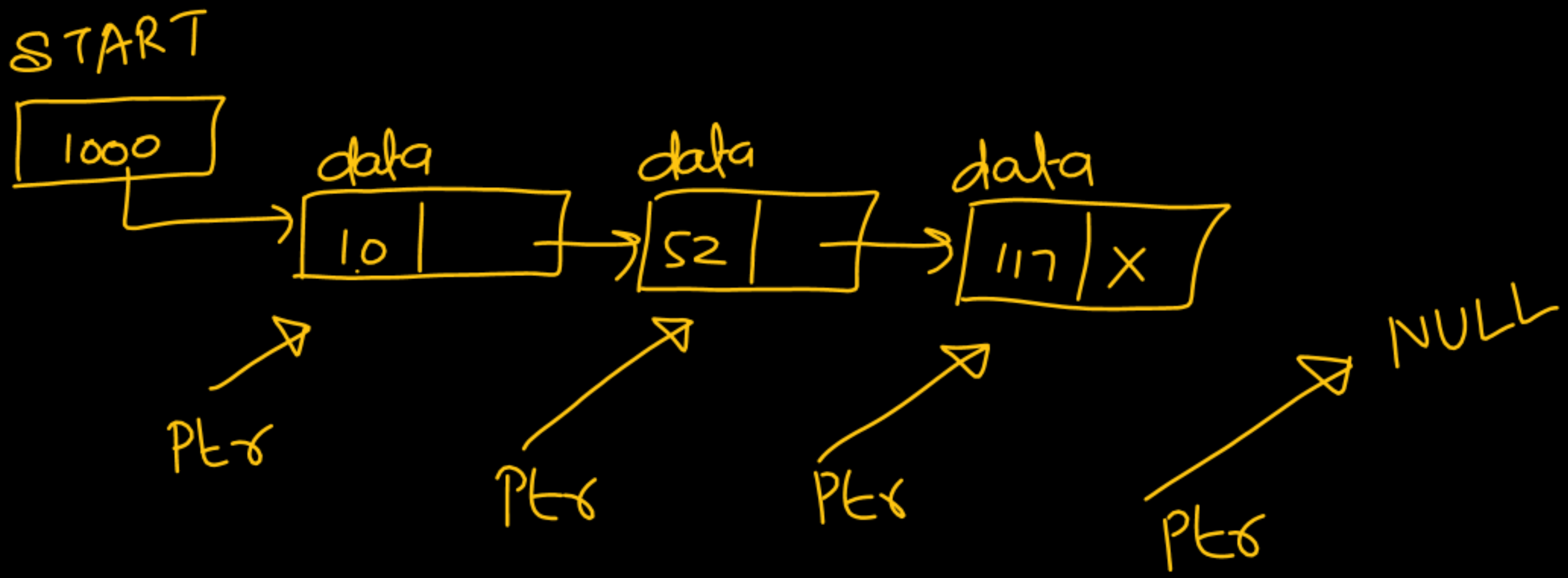
Key : 5

O/P : YES

```
void  Search( int Key)
{

}
```

```
void Search(struct Node *Ptr, int Key)
{

}
```

```
main( )
{
  Search(START,
         Key);
}
```



Key = 32

Key = 117

```
void Search(struct Node *Ptr, int key)
{
      While (Ptr != NULL)
      {
          if(Ptr -> data == Key)
          {
              printf("YES");
              return;
          }
          Ptr = Ptr -> next;
      }
 ⇒  printf("No");
}
```

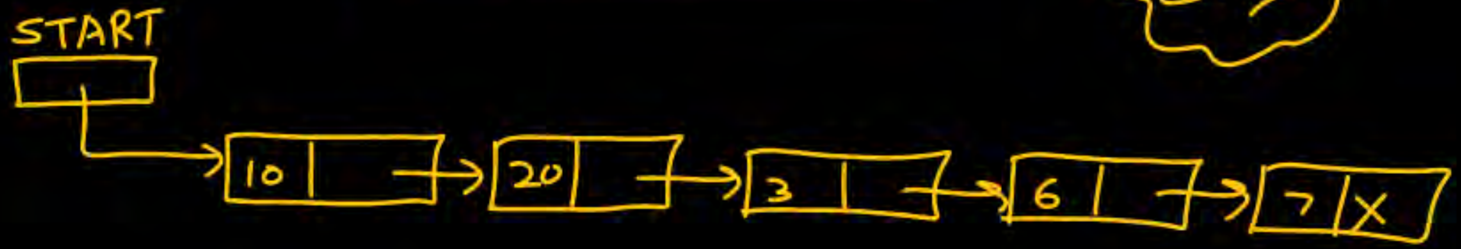
Ptr == NULL

4hr 4hr 4hr ------

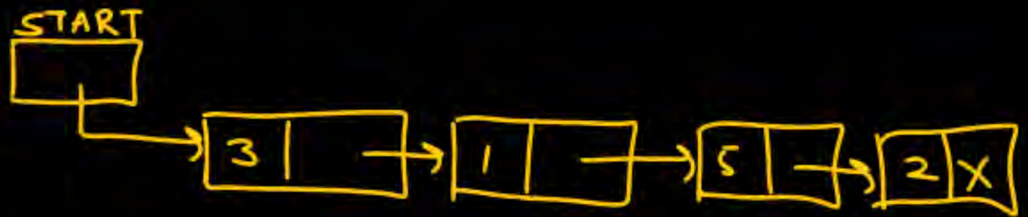
✗

Printing Alternates Nodes

Easy

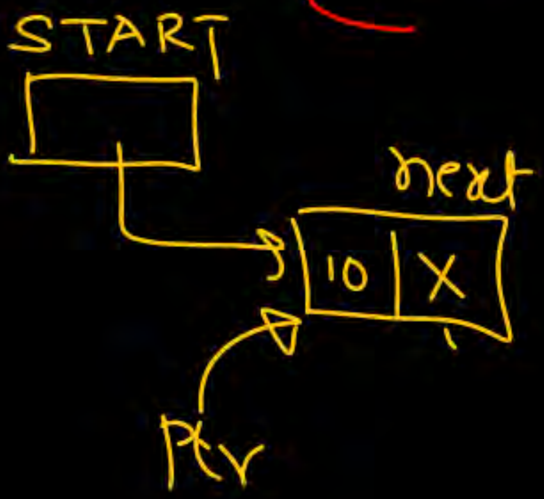


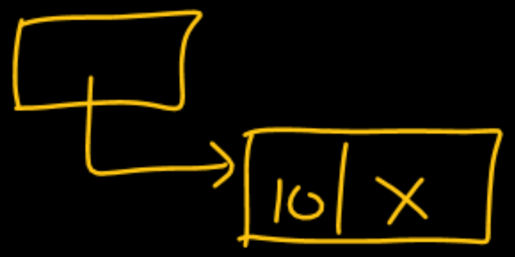
10 3 7



3 5

```
Ptr = START;

While (Ptr != NULL)
{
    printf("%d", Ptr->data);
    Ptr = Ptr->next->next;
}
```

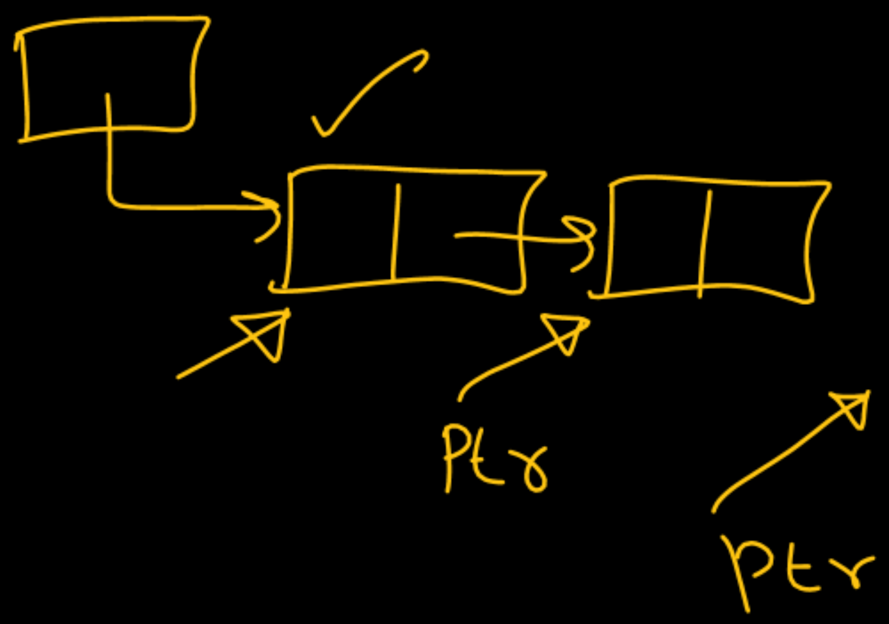




```
While(Ptr != NULL && Ptr->next != NULL)
  {
      pf( ptr->data);
      Ptr = Ptr->next->next;
  }
```

```
count = 0;

While (Ptr != NULL)
  {

      if (count % 2 == 0)
          printf("%d", Ptr->data);
              -   ;

          Ptr = Ptr->link;
          Count ++;

  }
```

count = 1



THANK YOU GW SOLDIERS !