CS & IT ENGINEERING

Data structures & Programming

Linked List

Lec- 06

By- Pankaj Sharma sir

Linked List -6

1. The following C function takes a S.L.L P as an argument.

```
int f( struct item *P){

    return ( (P==NULL) || (P->next==NULL)) ||
        (((P->data <= P->next->data) && f(P->next)));
}
```

the func. returns 1 $\boxed{\text{if and only if}}$

A.) The list is empty or exactly 1 ele.

B.) The ele in list are sorted in non-der. order.

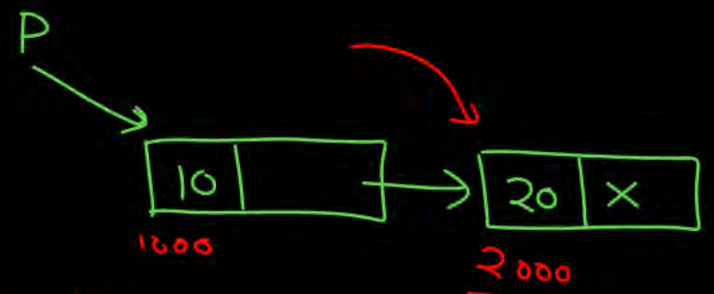
C.) The " " " " " " -inc "

D.) Not all the ele. in the list have same data.

1. The following C function takes a S.L.L P as an argument.

```
int f( struct item *P){
    return ( (P==NULL) || (P->next==NULL)) ||
           ((P->data <= P->next->data) && f(P->next)) ;
}
```

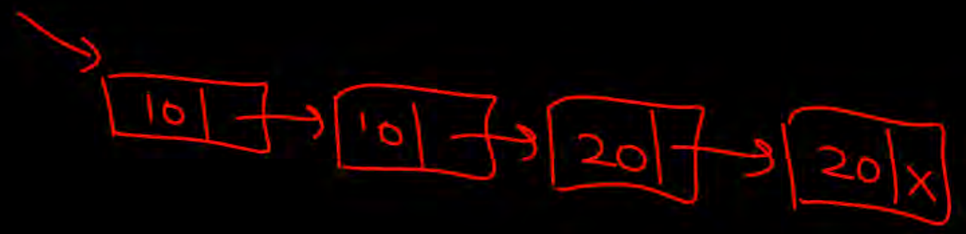
0 1



return 0 || 0 || (10 <= 20)

0 || 0 || 1 &&

return 1 && f(2000)



10, 10, 20, 20

(distinct ele)

Asc./Increasing → बढ़ता

1, 2, 5, 7, 8

(Duplicate ele)

Non-decreasing

1, 1, 1, 3, 5, 5

(2) In worst case, the no. of comparisions needed to search a singly linked list of length n for a given element is :-

A) $\log n$

B) $n/2$

C) $\log n - 1$

D) n

worst case :

3. what is the worst case time complexity to reverse a singly linked list in O(1) space ?

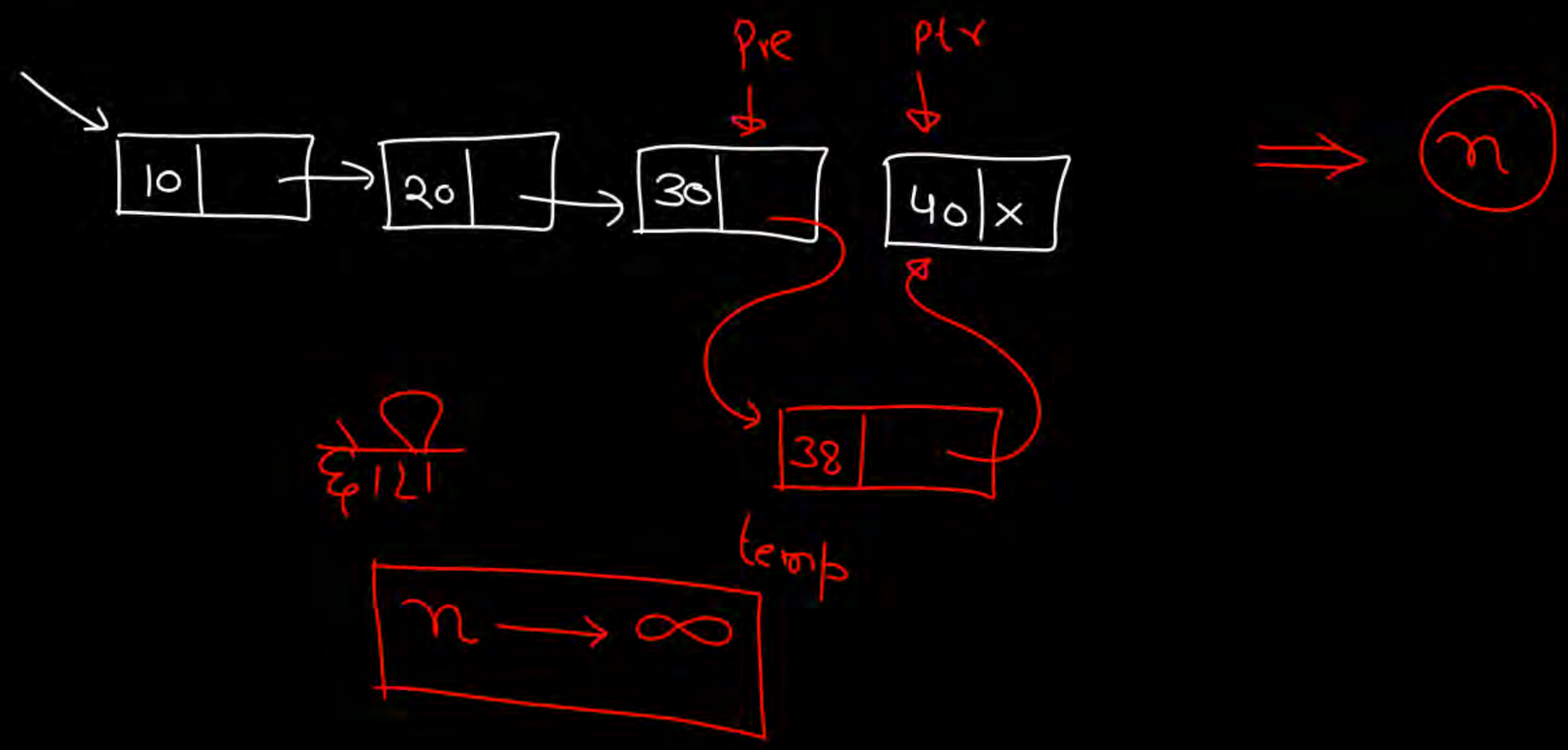
O(n)

Constant memory

1
2
3

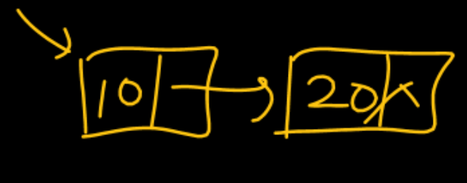
limited no. of var.

4. What is the $\boxed{\text{worst case}}$ T.C. of inserting n elements into an Empty l.l., if the l.l. needs to be maintained in sorted order.



$\Rightarrow n$

$n \longrightarrow \infty$



1 → 0

2nd → 1

3rd ele ⇒ 2 com

4th ⇒ 3 comp.

5th ⇒ 4 comp

⋮

nth ⇒ (n-1) comp

$1+2+3+\cdots(n-1)$

$$= \frac{(n-1)(n)}{2}$$

$$= \frac{n^2}{2} - \frac{n}{2}$$

```
void join(node *m, node *n){
    node *p = n;
    while(p->next != NULL)
    {
        p = p->next;
    }
    p->next = m;
}
```

Assuming that m and n points to valid NULL terminated LL. Invocation of join will.

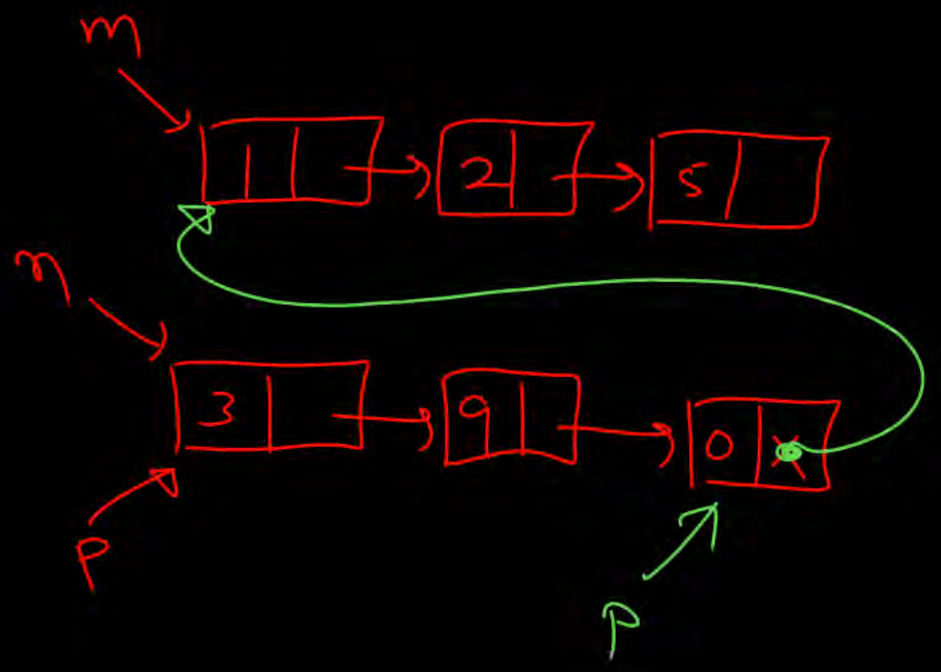
A) append list m to the end of list n for all i/ps

B) Either cause a null pointer dereference or append list m to the end of list n

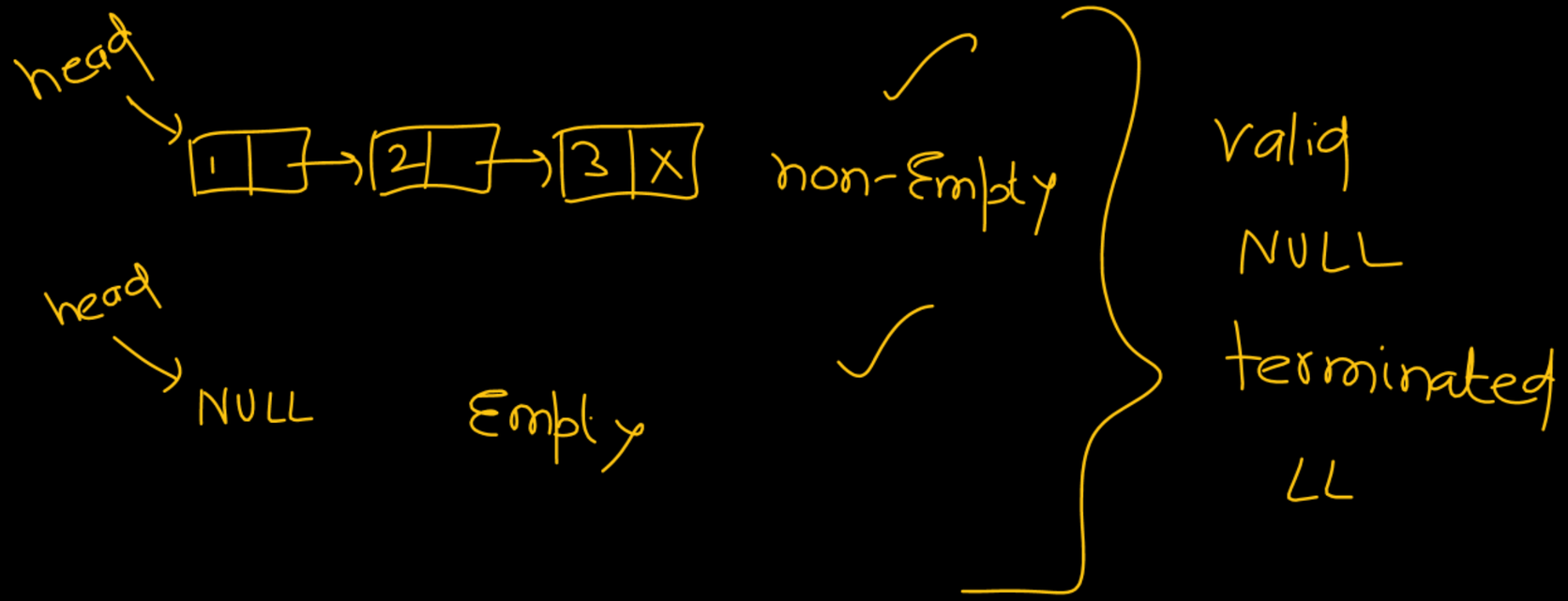
C) Cause a null pointer derefrence.

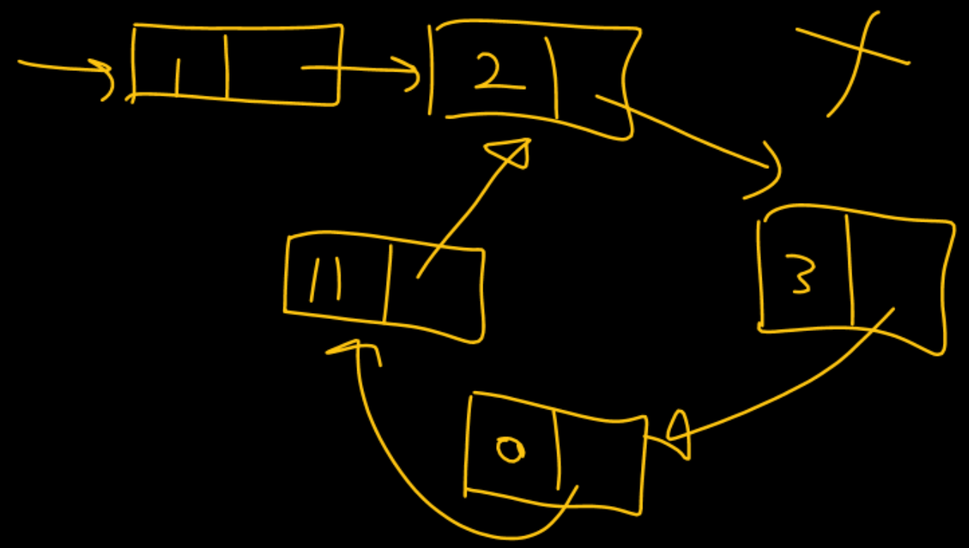
d) Append list m to the end of list n.

```
void join( node *m, node *n){
        node *p = n;
        while (p->next != NULL)
            {
                p = p->next;
            }
        p->next = m;
    }
```



Assuming that m and n points to valid NULL terminated LL.
Invocation of join will.


head
1
2
3
X
non-Empty
head
NULL
Empty
valid
NULL
terminated
LL


1
2
11
3
0

7) rearrange the ele. $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

5 min

```
void rearrange(struct node *list)
{
    struct node *p, *q;
    int temp;
    if(!list || !list->next) return;
    p = list; q = list->next;
    while(q) {
        temp = p->value;
        p->value = q->value;
        q->value = temp;
        p = q->next;
        q = p ? p->next : 0;
    }
}
```

list == NULL

list->next == NULL

7) rearrange the ele.

2 1 4 3 6 5

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 7$

NULL

5 min

2 1 4 3 6 5 7

p q

```
p = list; q = list->next;
while(q) {
        temp = p->value;
        p->value = q->value;
        q->value = temp;
        p = q->next;
        q = p ? p->next : 0;
}
```

swap

True/non-zero

q = (p) ? p->next : 0;

LL-7 08:30 PM

- Types of LL
- PYQs


THANK YOU GW
SOLDIERS !