Linked list: Implementation

Course on C-Programming & Data Structures: GATE - 2024 & 2025

Vishvadeep Gothi • Lesson 28 • Dec 27, 2022

Data Structure:
Linked List 2

By: Vishvadeep Gothi

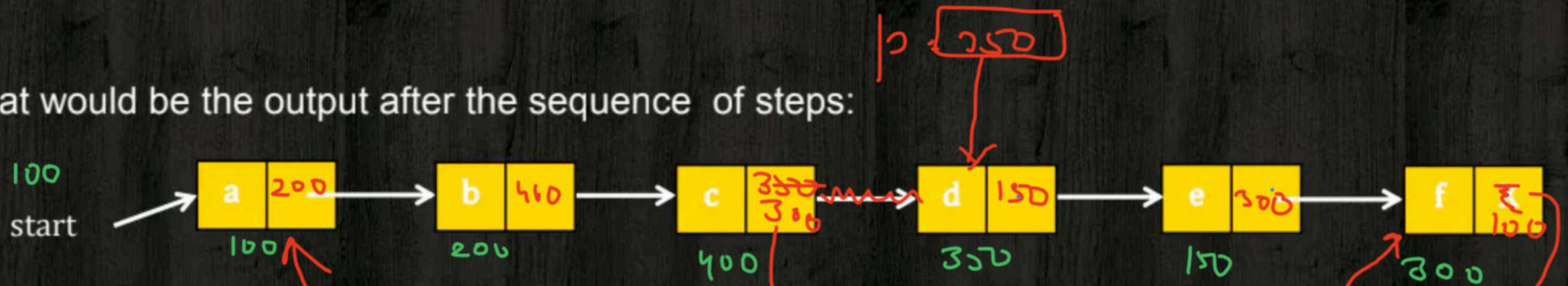
Hello!

I am Vishvadeep Gothi

I am here because I love to teach

Practice Question 1

What would be the output after the sequence of steps:



struct node *p;

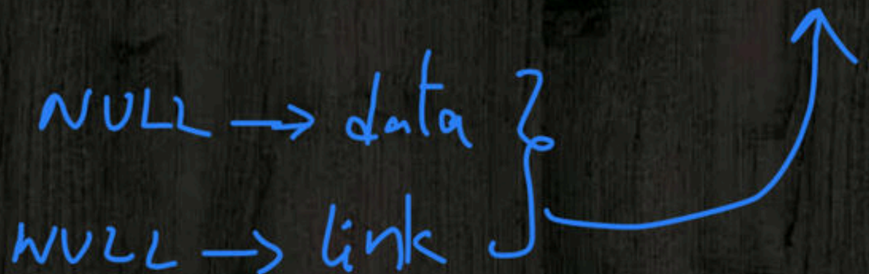
p =start→ link → link → link;

start → link → link → link = p → link → link;

p → link → link → link = start;

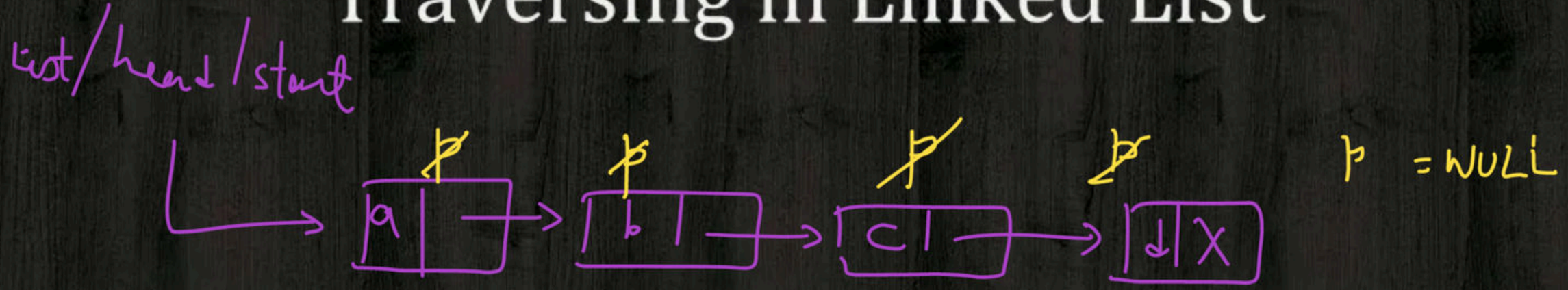
printf("%c", start → link → link → link → link → data); a

NULL Pointer Dereference




```
p = ------ ;
if (p → link != NULL)
```

Traversing in Linked List




```
struct node *p = start;
while (p != NULL) or while(p)
{
  process p -> data;
  p = p -> link;
}
```

abcd

R.T.
complexity = O(n)

Number of Elements in Linked List

node

```
struct node *p = start;
int count = 0;
while (p)
{
  count++;
  p = p -> link;
}
return count;
```

R.T. Complexity = $\Theta(n)$

Consider a linked-list with integer elements. Return sum of all elements of list.

```
struct node *p = start;
int sum = 0;
while(p)
{
  sum = sum + p->data;
  p = p->link;
}
```

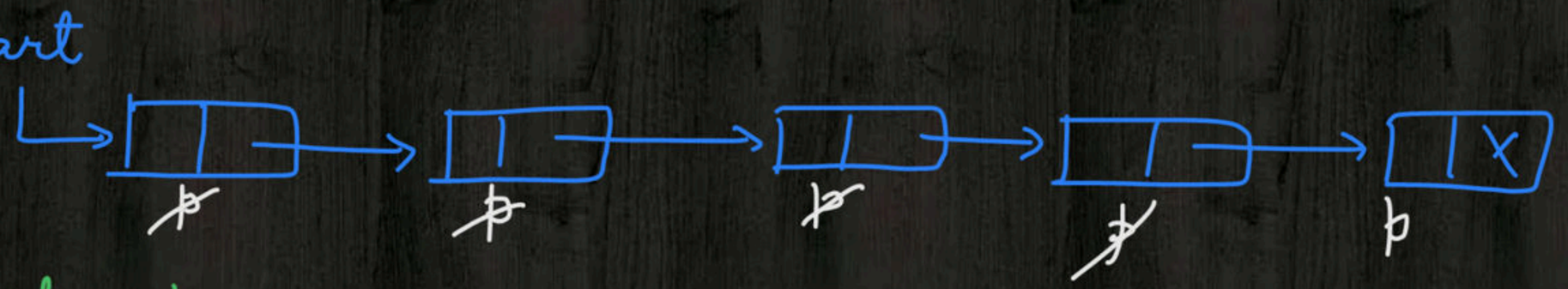
$O(n)$

Find min of linked-list ?

```
if (start == NULL) return 0;
int min = start -> data;
struct node *p = start -> link;
while (p)
{
   if ( p -> data < min)
    {
     min = p -> data;
    }
   p = p -> link;
} return min;
```

$\theta(n)$

Address of Last Node in Linked List




```
struct node *p = start;
if (p == NULL) return NULL;
while (p -> link != NULL)
{
   p = p -> link;
}
return p;
```

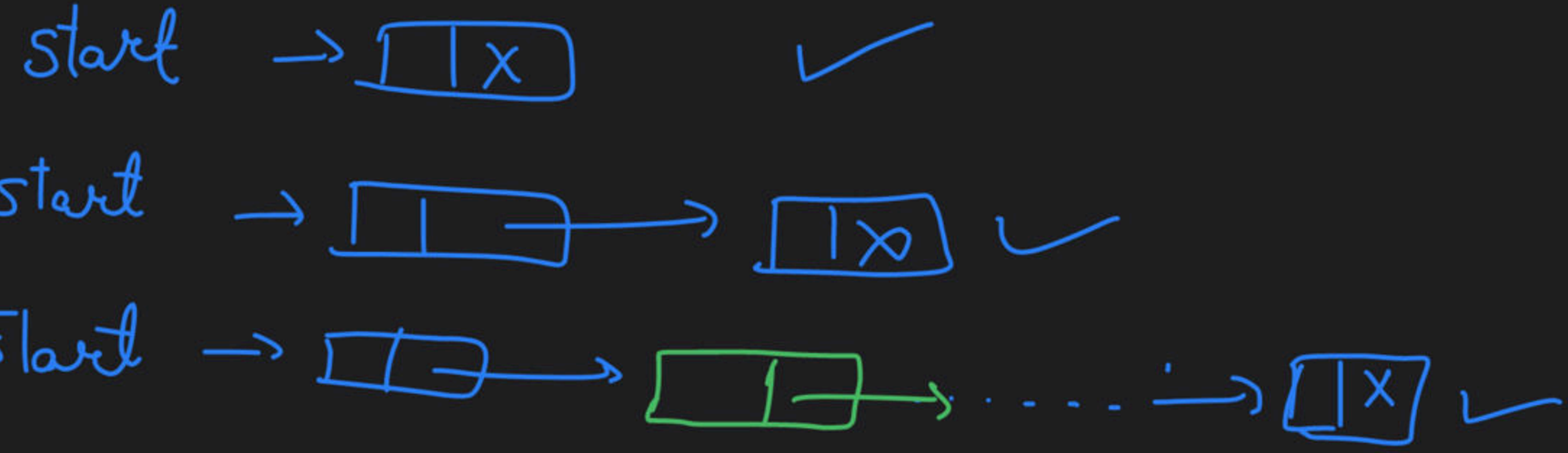
or

```
while (p -> link)
```

$O(n)$

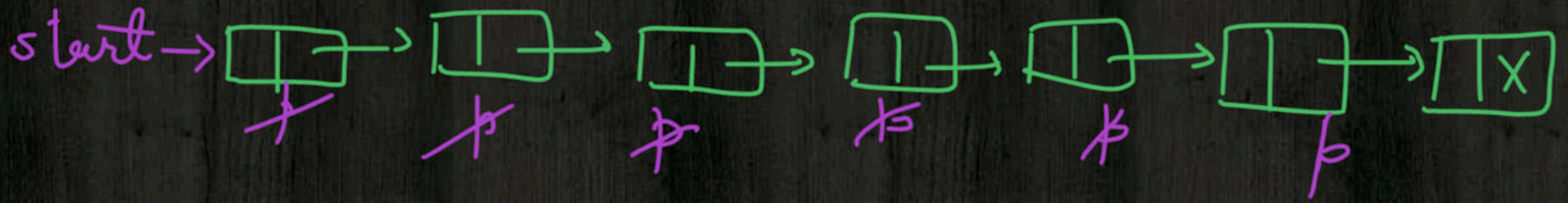
Valid NULL Terminated linked-list :-

ex:-



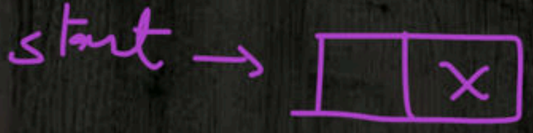
Question

What would be the output after the following code segment is executed on a valid NULL terminated singly linked list.




```
struct node *p;
p = start ;
while(p → link → link)
{
p  = p→ link;
}
printf("%c", p → data );
```

start = NULL



Ans: either NPD or data of second last node.

```
if (p == NULL || p->link == NULL)
    return
```

or `if (!p || !p->link)`

```
while (p->link->link)
```

Insertn in Linked_list

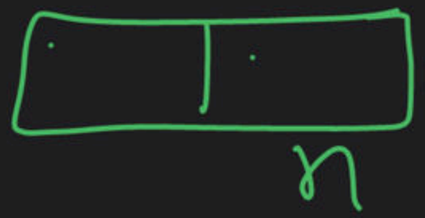
can be done anywhere according to requirement.

whenever a new node to be inserted

- ① First create a new node dynamically using malloc()
- ② Insert this new node.

Creating a new node:-

```
struct node *n = (struct node *)malloc(sizeof (struct node));
```

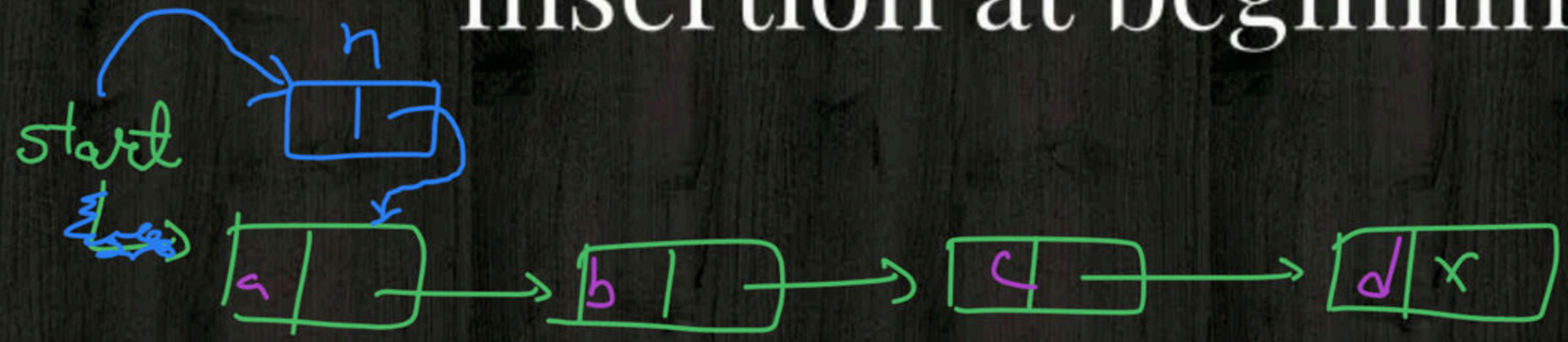



```
if (n==NULL)
  {
    no memory left
  }
```

Insertion in Linked List

1. Insertion at beginning
2. Insertion after given node
3. Insertion at the end

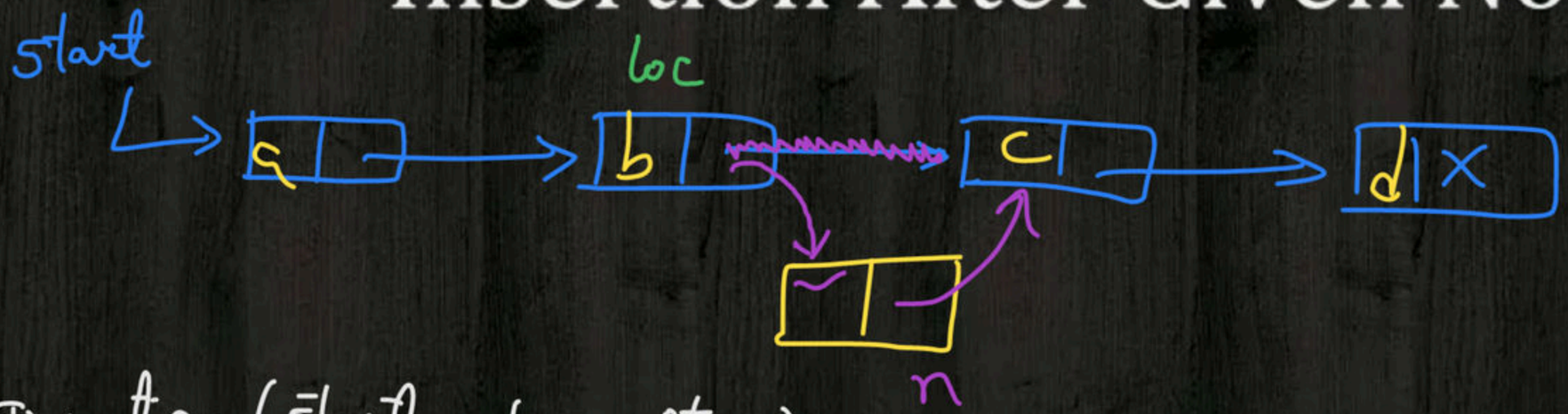
Insertion at beginning




```
Insertion (start, item)
{
  n → data = item
  n → link = start
  start = n;
}
```

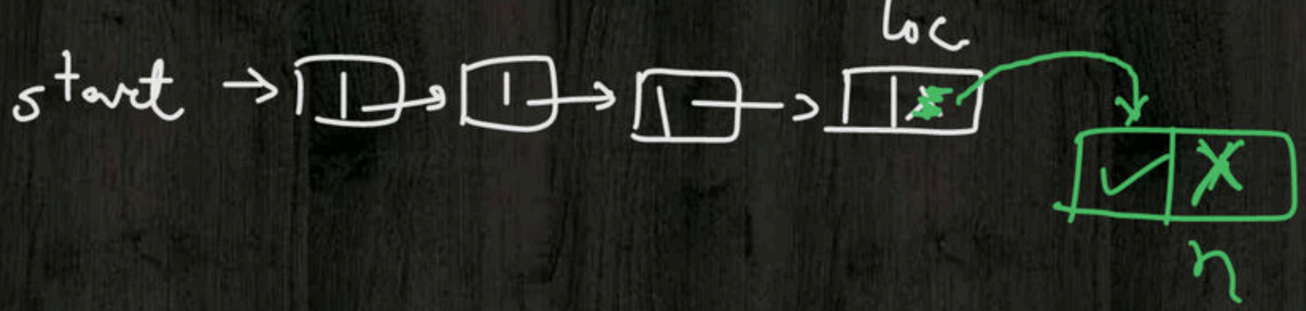
R. T. Complexity = $\Theta(1)$

Insertion After Given Node

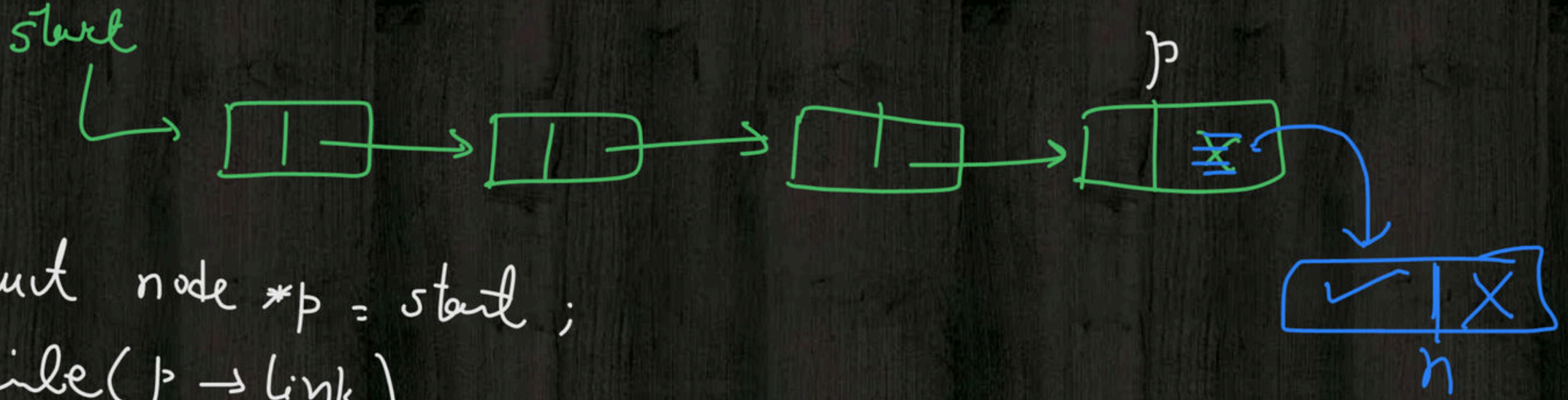



```
Insertion (start, loc, item)
{
  n -> data = item
  n -> link = loc -> link
  loc -> link = n
}
```

R.T. complexity = $\Theta(1)$



Insertion At the End

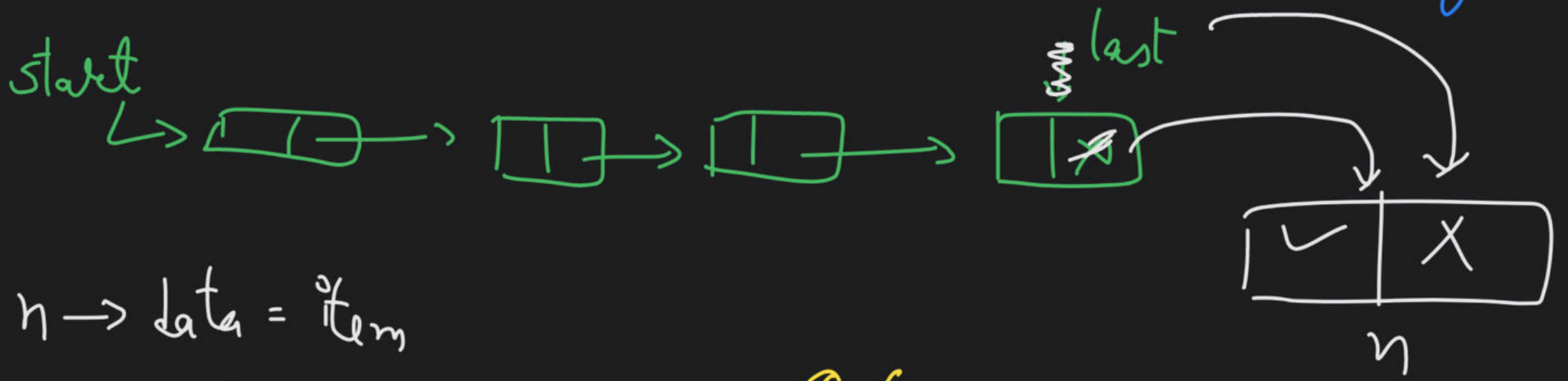



```
struct node *p = start;
while(p -> link)
{
  p = p->link
}
```

n -> data = item
n -> link = NULL
p -> link = n

R.T. complexity = $\Theta(n)$

Insertn at the end when addr. of last node is given



n → data = item

n → link = NULL

last → link = n

last = n

$\theta(1)$

Deletion in Linked List

1. Deletion at beginning
2. Deletion of given node
3. Deletion at the end

Deletion At Beginning

Deletion Of a Given Node

Deletion At the End

Question 1 GATE-2004

Suppose each set is represented as a linked list with elements in arbitrary order. Which of the operations among union, intersection, membership and cardinality will be the slowest?

(A) Union only

(B) Intersection, membership

(C) Membership, cardinality

(D) Union, intersection

Application of Linked List

- To store polynomials
  - Univariate
  - Bivariate

DPP

Question 1

Write an algorithm to convert the list into a circular list?

Question 2

Following C- like function takes a singly-linked list of integers as a parameters and rearranges the elements of list. The function is called with the list containing the integers 3, 5, 5, 5, 7, 8, 9, 9, 9, 9, 12, 15, 18, 19, 23 in the given order. What will be the total no. of contents of the list after the function completes execution (no. of nodes in the list)

```
struct node
 {
 int data;
struct node * next;
};
```

Question 2

```
void rearrange (struct node * head)
{
struct node * current = head;
if( current = = NULL) return;
while (current →next ! = NULL)
{
   if (current →data = = current → next→ data)
      {
      struct node *P = current → next→next;
      free (current → next );
      current → next=p;
      }
   else
   {
      current = current → next;
   }}}
```

Question 3

Consider the following function on a valid NULL terminated list:

```
int func(node *start)
struct node *p;
p = start ;
while(p → link )
{
p  = p→ link;
}
return 1;
```

The above function returns:

(A) 1 always

(B) None always

(C) Will cause error always

(D) Error or returns 1

Question 4

There is node pointer x, which points to a node of a singly linked list. Delete the node after the node pointed by pointer x.

Question 5

There is node pointer x, which points to a node of a singly linked list. Delete the node before the node pointed by pointer x.

Question 6

What would be the output after the sequence of steps:

struct node *p;

p =start→ link → link;

start → link → link → link = p → link → link;

p → link → link → link = start;

printf("%c", start → link → link → link → link → data);

Question 7

What would be the output after the sequence of steps:

struct node *p, *q;

p =start → link → link;

q = p → link → link;

p → link = q → link;

q → link → link = start;

printf("%c", start → link → link → link → link → data);

Question 8

What would be the output after the sequence of steps:

struct node *p, *q;

p =start → link;

q = p → link → link → link;

p → link = q → link → link;

q → link → link = p;

printf("%c", start → link → link → link → link → data);

Happy Learning