CCBP 4.0

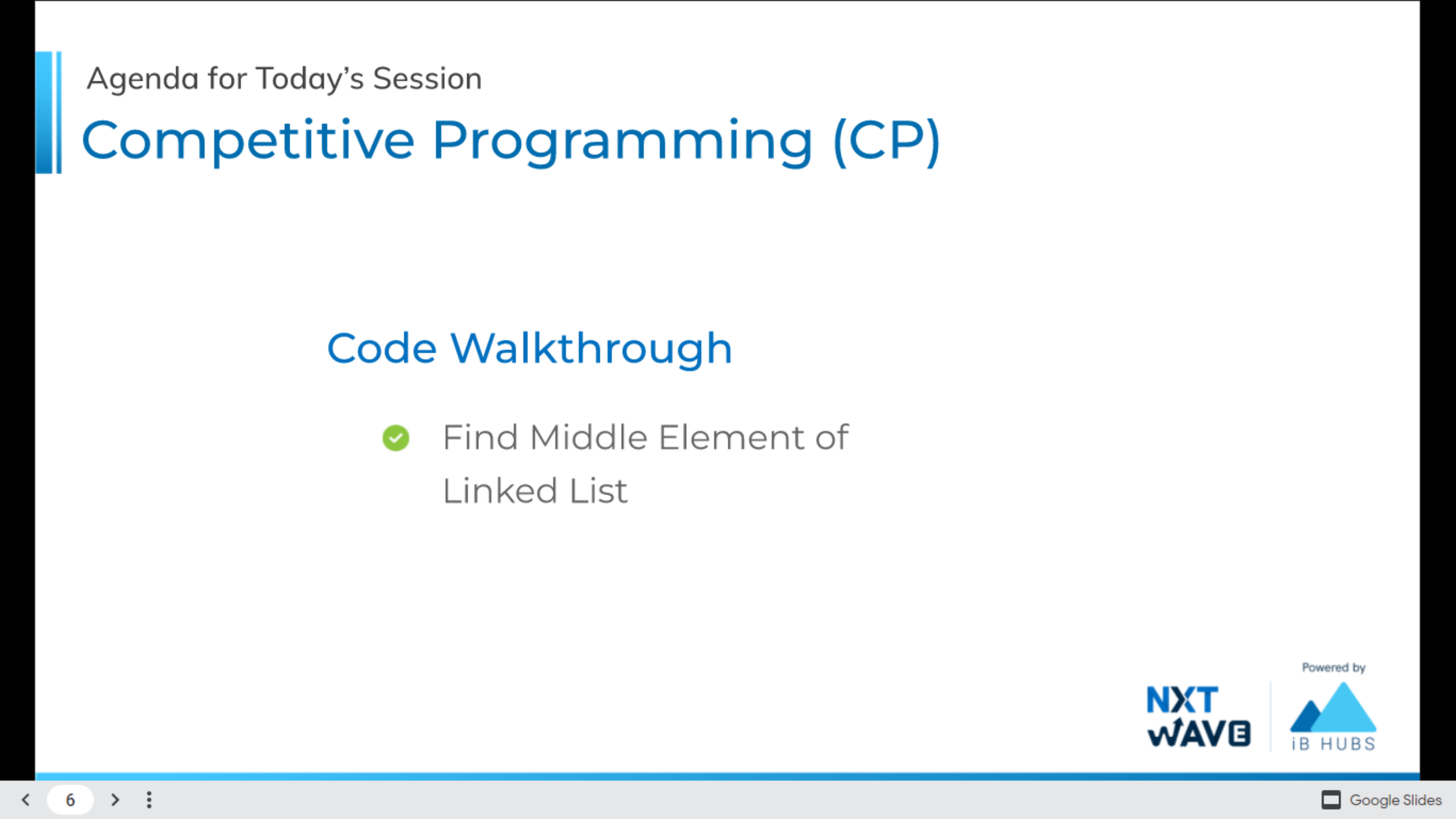
21st Century Skills

# Linked Lists
# Code Walkthrough - 1

Powered by

NXT WAVE

iB HUBS



Agenda for Today's Session

# Competitive Programming (CP)

## Code Walkthrough

- ✓ Find Middle Element of Linked List

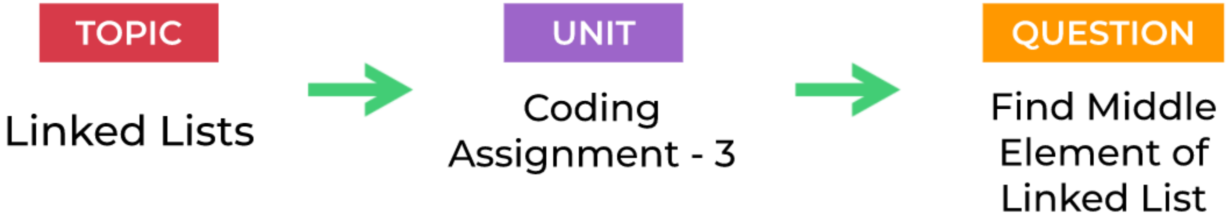
Powered by

NXT WAVE

iB HUBS

Code Walkthrough

# Find Middle Element of Linked List



TOPIC

Linked Lists

UNIT

Coding Assignment - 3

QUESTION

Find Middle Element of Linked List

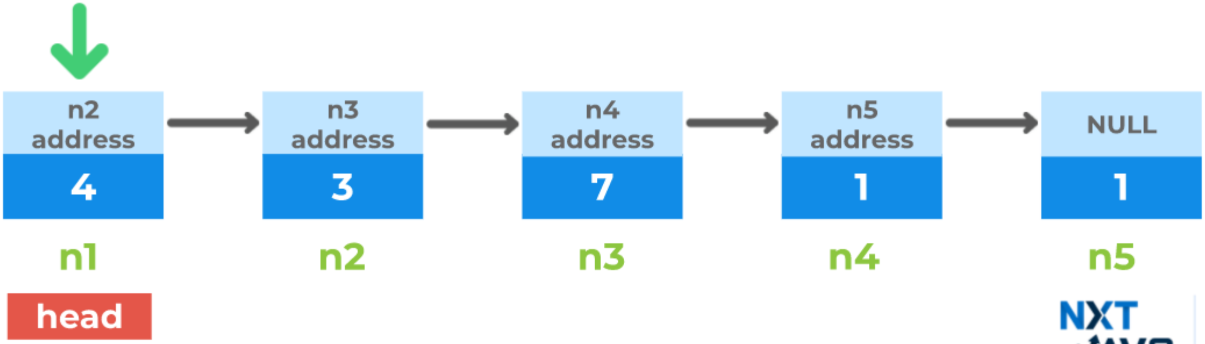
Powered by NXT WAVE iB HUBS

Find Middle Element of Linked List

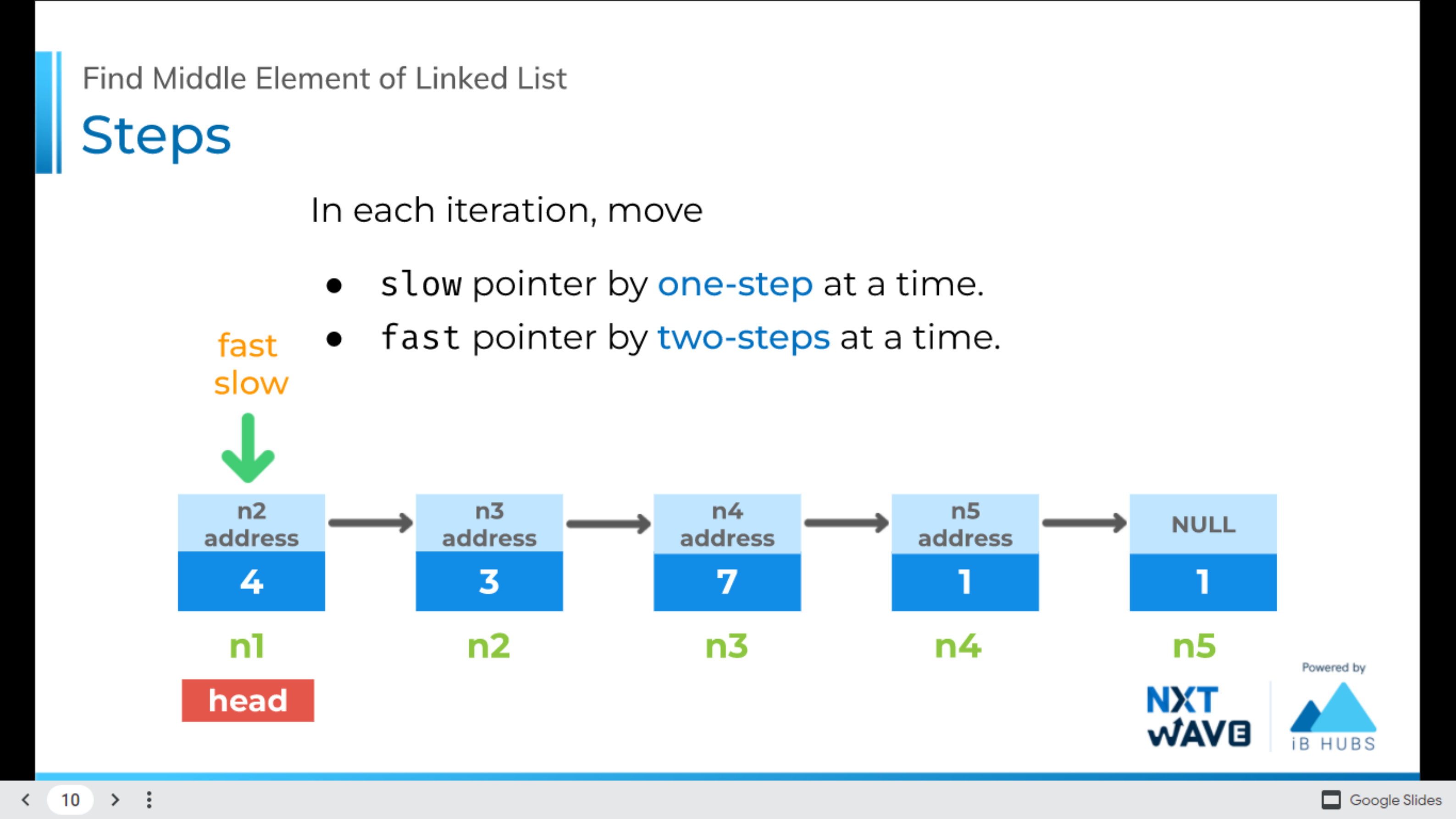
# Steps

Initialize two pointers slow and fast which are pointing to the head.

fast
slow



| n2 address | n3 address | n4 address | n5 address | NULL |
|---|---|---|---|---|
| 4 | 3 | 7 | 1 | 1 |
| n1 | n2 | n3 | n4 | n5 |
| head | | | | |



Find Middle Element of Linked List

## Steps

In each iteration, move

- `slow` pointer by one-step at a time.
- `fast` pointer by two-steps at a time.

fast
slow

| n2 address | n3 address | n4 address | n5 address | NULL |
|---|---|---|---|---|
| 4 | 3 | 7 | 1 | 1 |
| n1 | n2 | n3 | n4 | n5 |

head

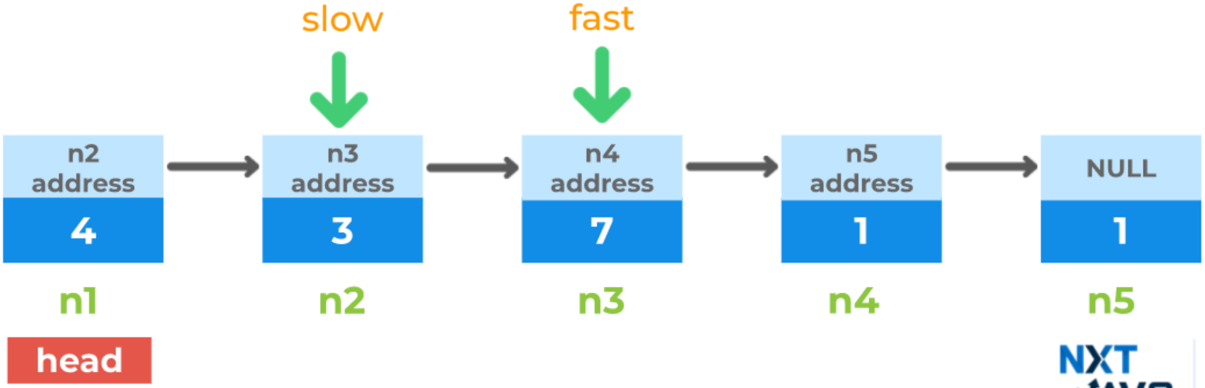
Powered by NXT WAVE iB HUBS

Find Middle Element of Linked List

# Steps

In each iteration, move

- `slow` pointer by one-step at a time.
- `fast` pointer by two-steps at a time.



slow

fast

| n2 address | n3 address | n4 address | n5 address | NULL |
|---|---|---|---|---|
| 4 | 3 | 7 | 1 | 1 |
| n1 | n2 | n3 | n4 | n5 |

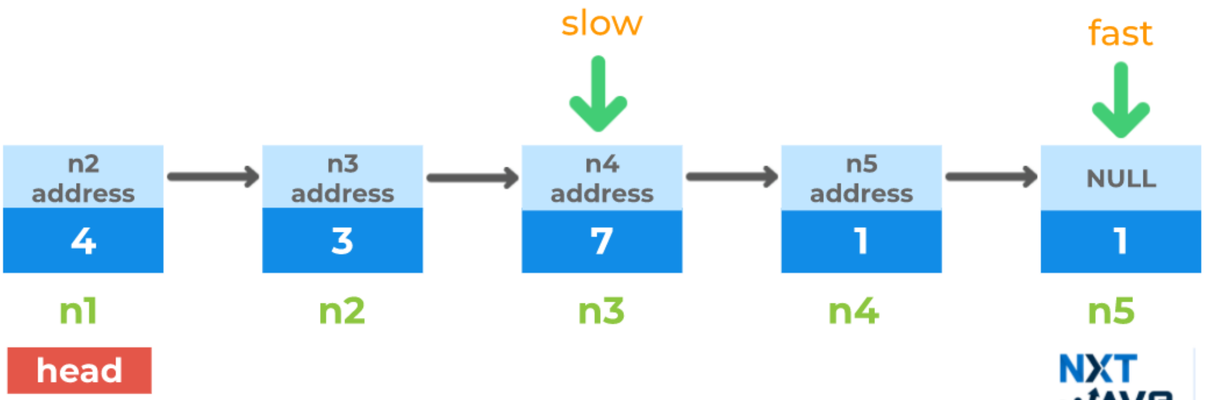
head

Find Middle Element of Linked List

# Steps

We stop iterating when

```
fast == NULL or fast→next == NULL
```



slow

fast

n2 address 4

n3 address 3

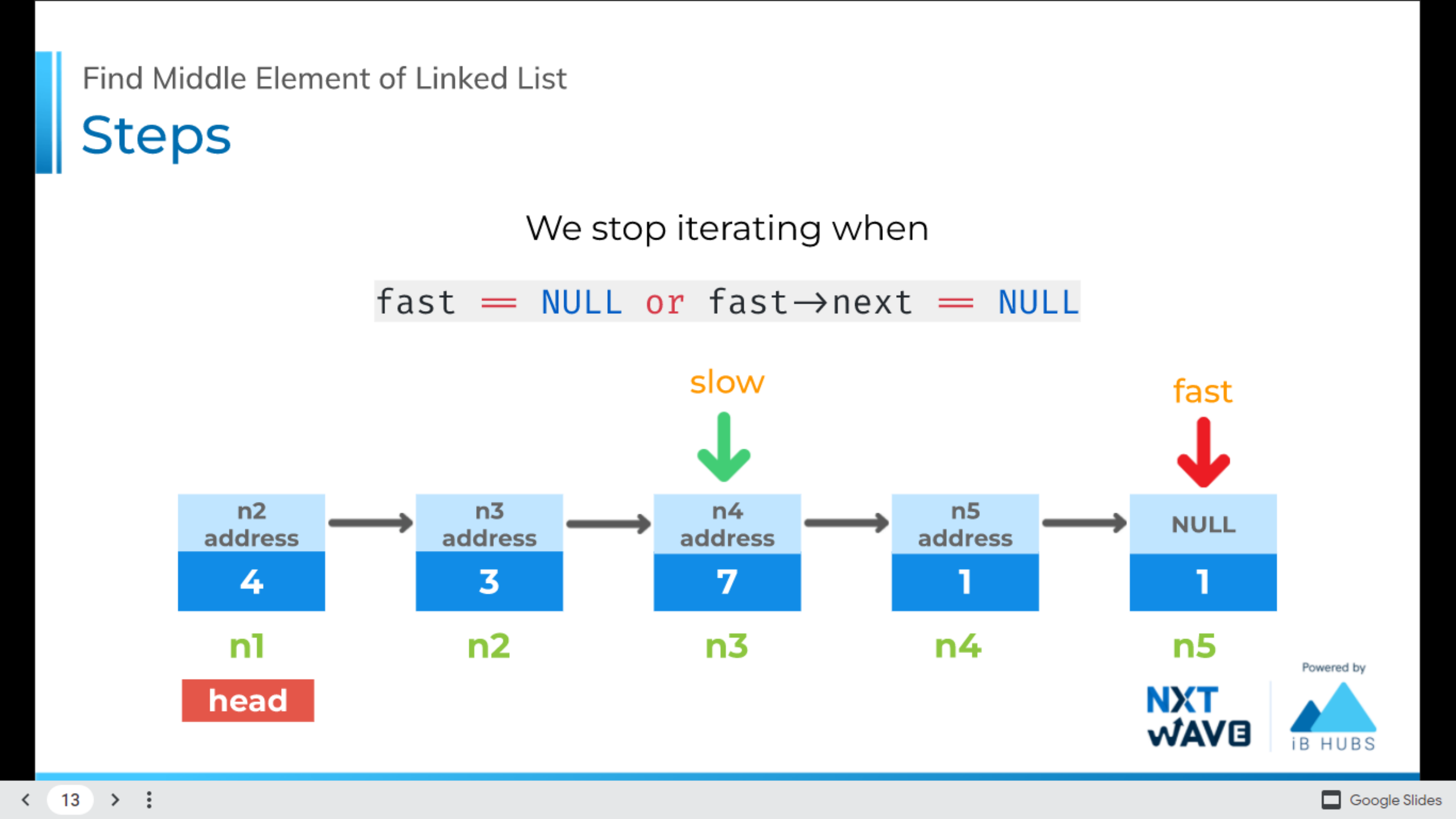
n4 address 7

n5 address 1

NULL 1

n1 n2 n3 n4 n5

head

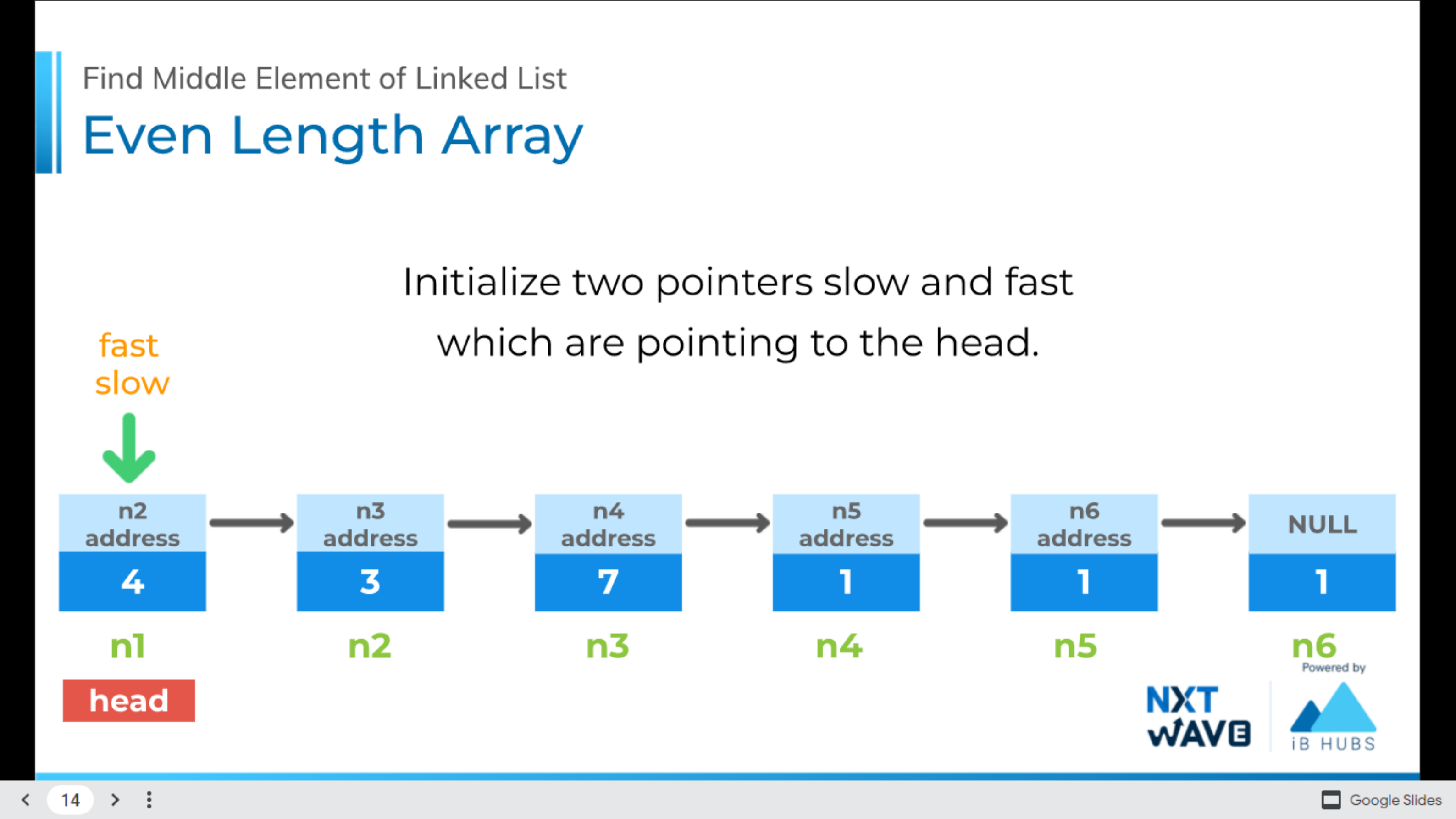


Find Middle Element of Linked List

## Steps

We stop iterating when

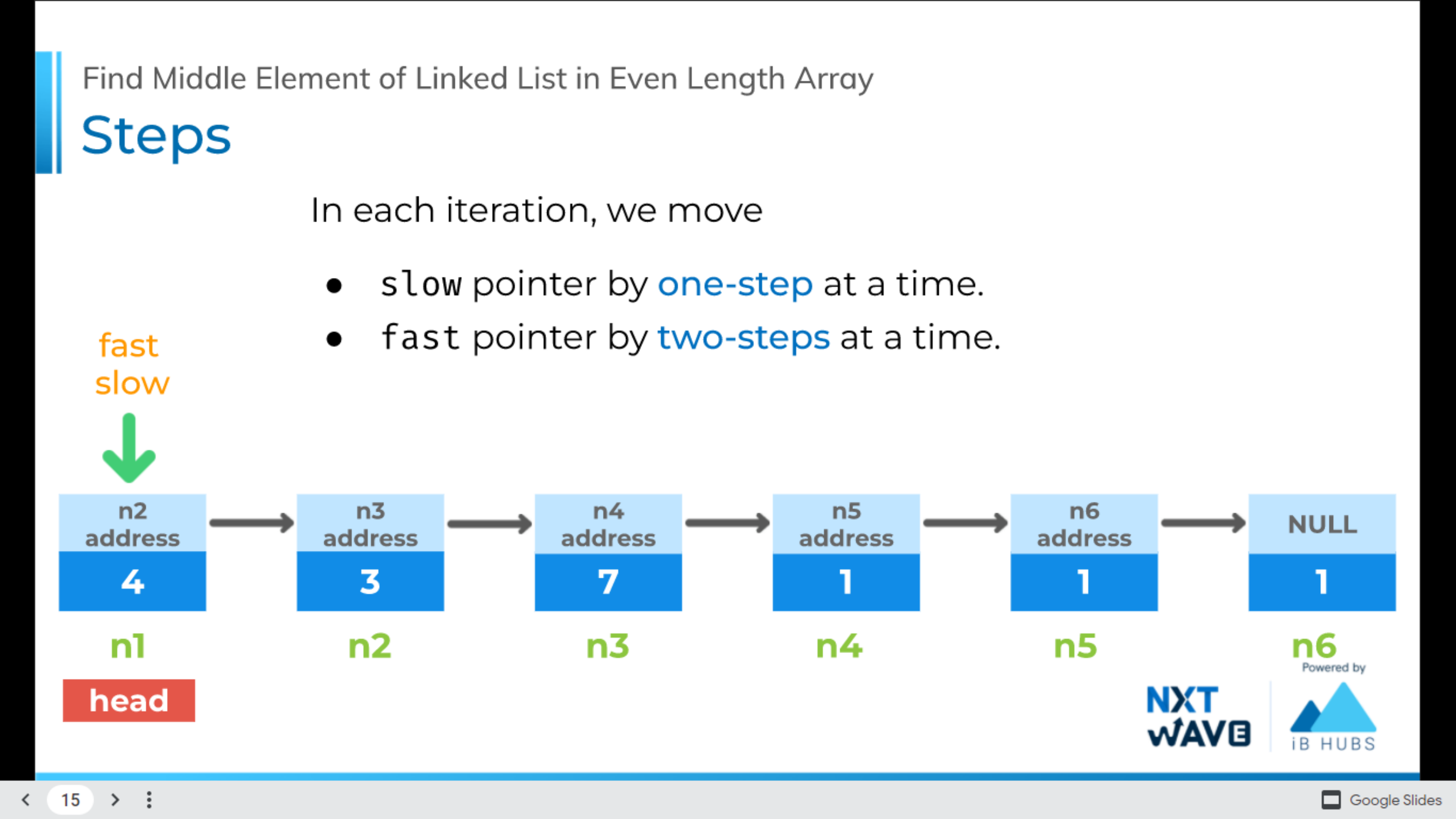
```
fast == NULL or fast→next == NULL
```



Find Middle Element of Linked List

# Even Length Array

Initialize two pointers slow and fast which are pointing to the head.



Find Middle Element of Linked List in Even Length Array

## Steps

In each iteration, we move

- `slow` pointer by one-step at a time.
- `fast` pointer by two-steps at a time.

fast
slow

| n2 address | n3 address | n4 address | n5 address | n6 address | NULL |
|---|---|---|---|---|---|
| 4 | 3 | 7 | 1 | 1 | 1 |
| n1 | n2 | n3 | n4 | n5 | n6 |

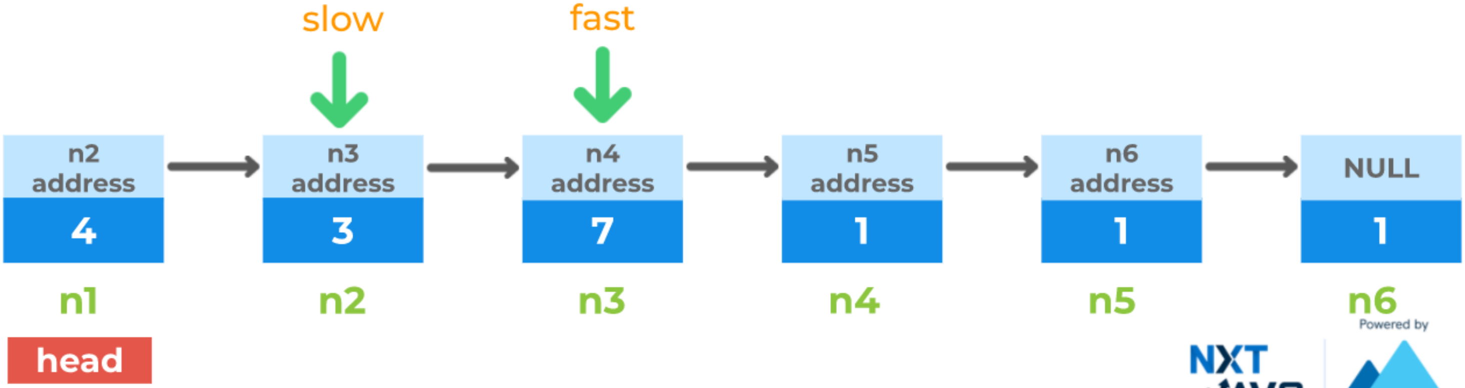
head

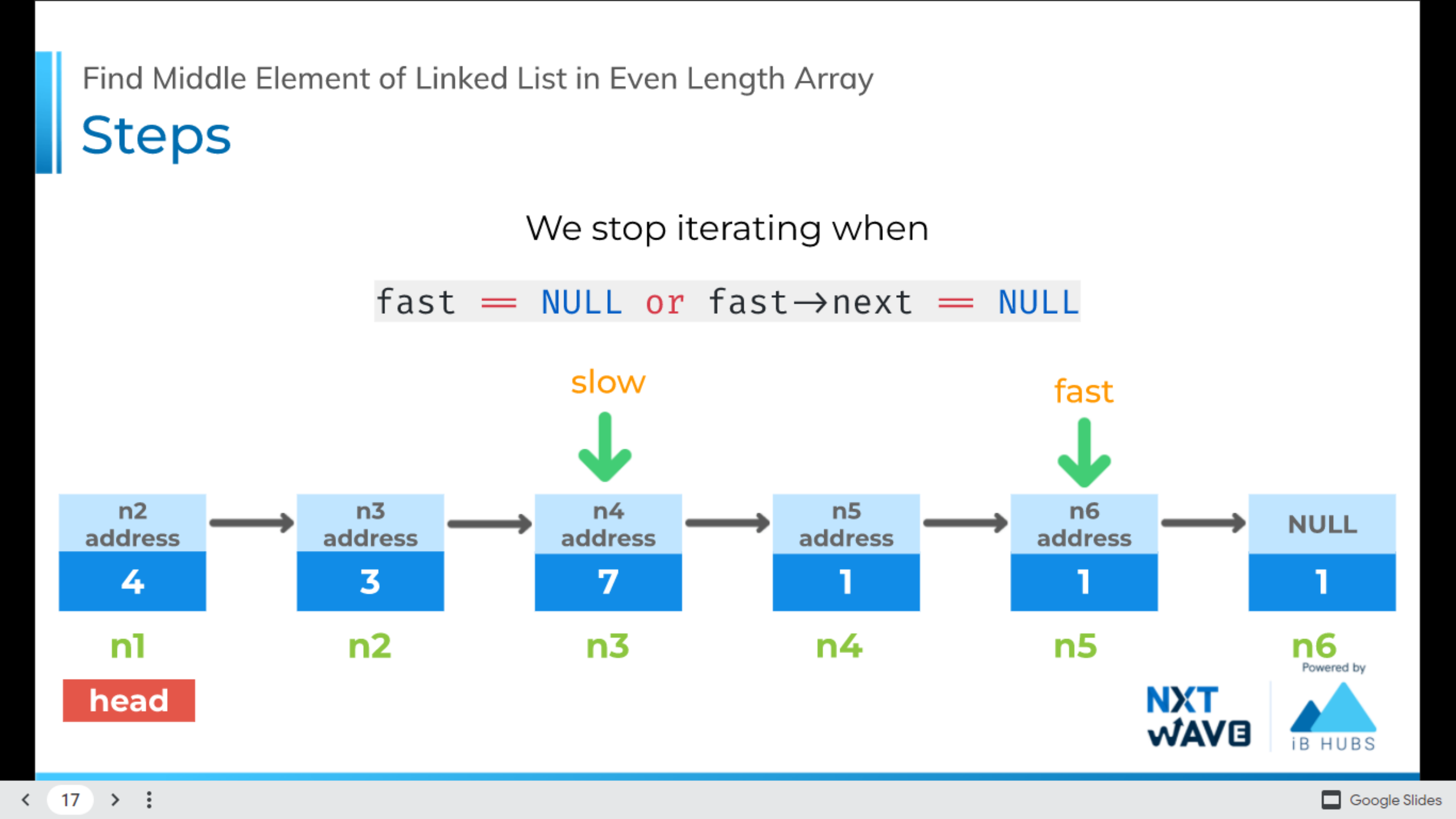
Find Middle Element of Linked List in Even Length Array

## Steps

In each iteration, we move

- `slow` pointer by one-step at a time.
- `fast` pointer by two-steps at a time.



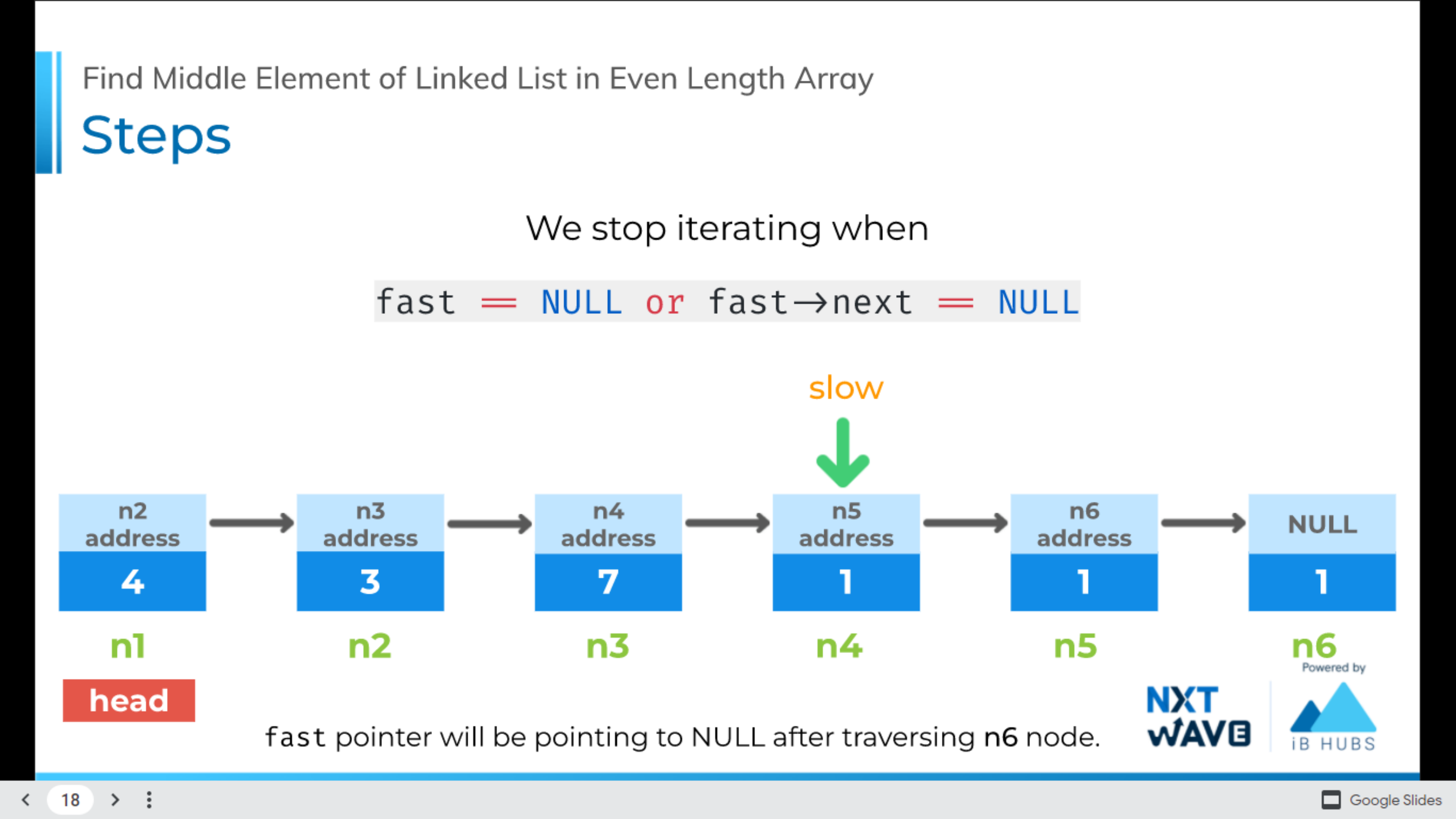


Find Middle Element of Linked List in Even Length Array

## Steps

We stop iterating when

```
fast == NULL or fast->next == NULL
```



Find Middle Element of Linked List in Even Length Array

## Steps

We stop iterating when

```
fast == NULL or fast->next == NULL
```

slow

| n2 address | n3 address | n4 address | n5 address | n6 address | NULL |
|---|---|---|---|---|---|
| 4 | 3 | 7 | 1 | 1 | 1 |
| n1 | n2 | n3 | n4 | n5 | n6 |

head

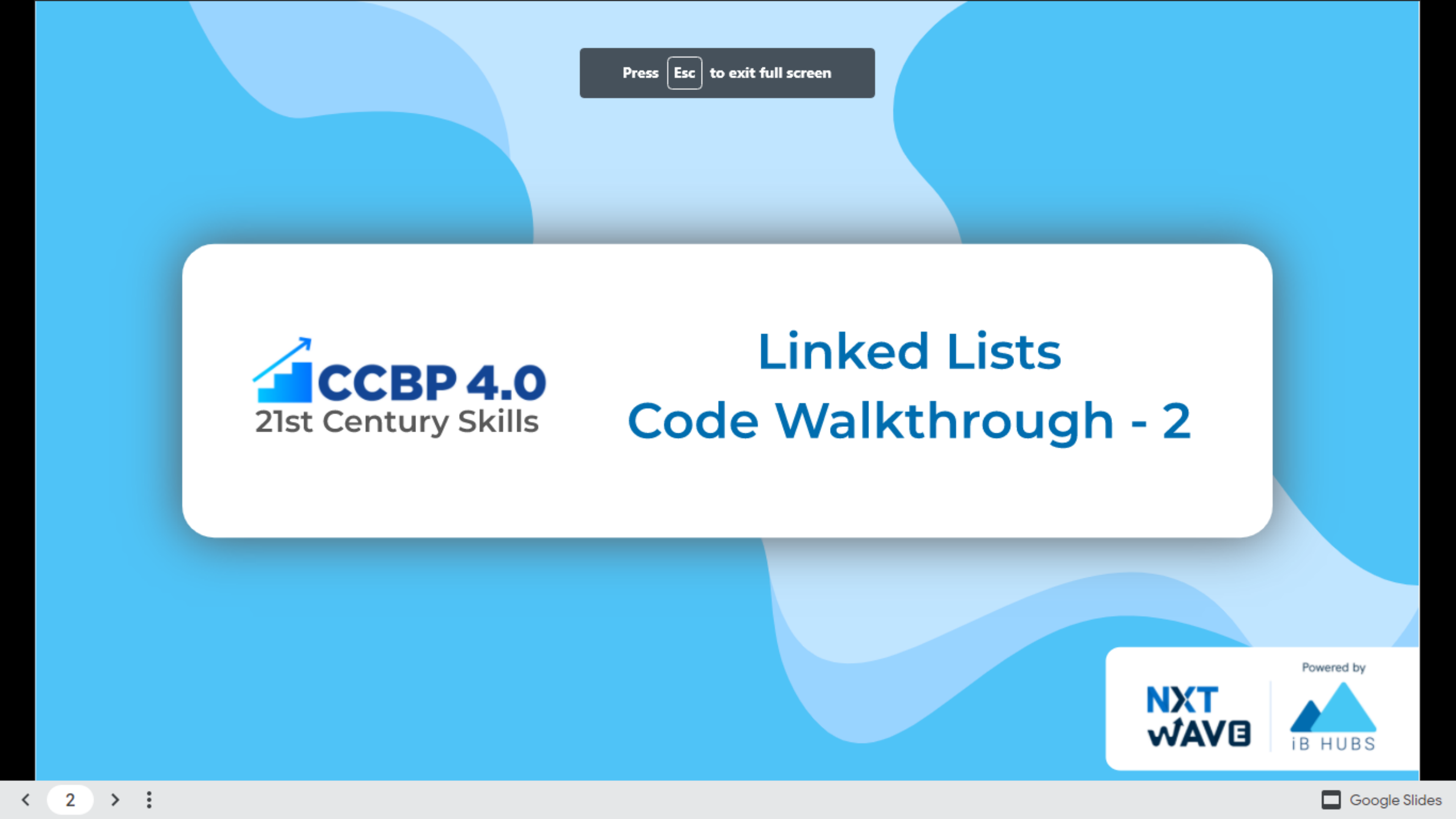
fast pointer will be pointing to NULL after traversing **n6** node.

# Key Takeaways

## Code Walkthrough

- ✓ Find Middle Element of Linked List

Powered by



CCBP 4.0
21st Century Skills

# Linked Lists
# Code Walkthrough - 2

Powered by
NXT WAVE
iB HUBS

Agenda for Today’s Session

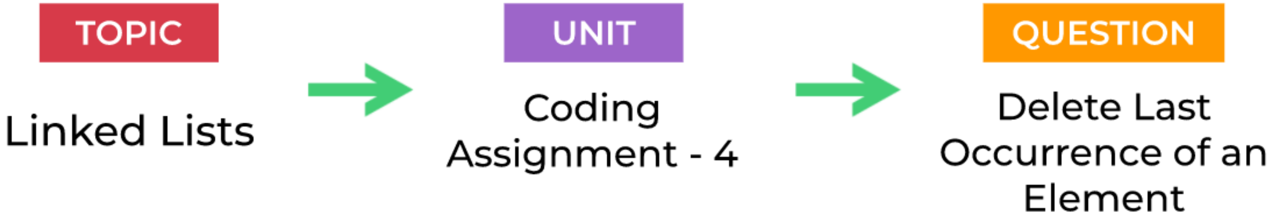
# Competitive Programming (CP)

## Code Walkthrough

- ✓ Delete Last Occurrence of an Element

Code Walkthrough

# Delete Last Occurrence of an Element



TOPIC

Linked Lists

UNIT

Coding Assignment - 4

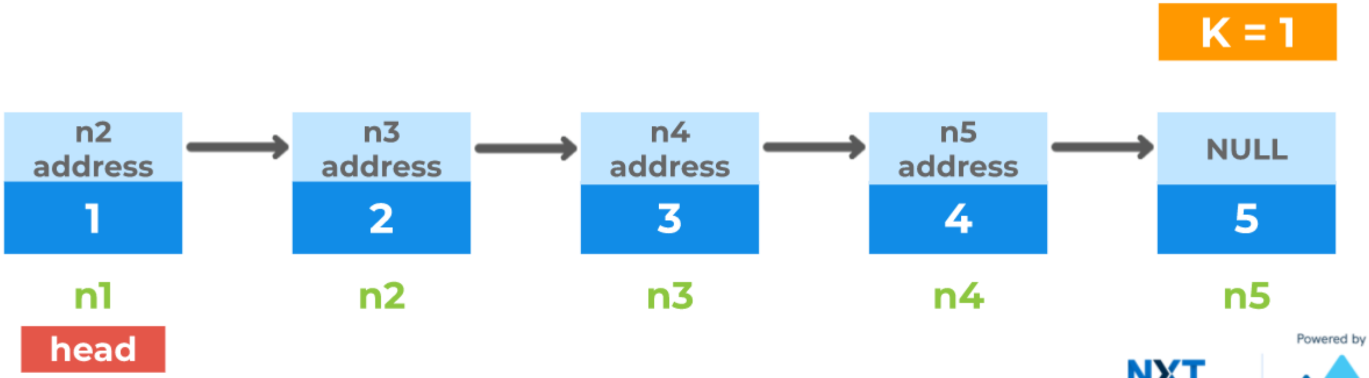
QUESTION

Delete Last Occurrence of an Element

Delete Last Occurrence of an Element

# Case - 1

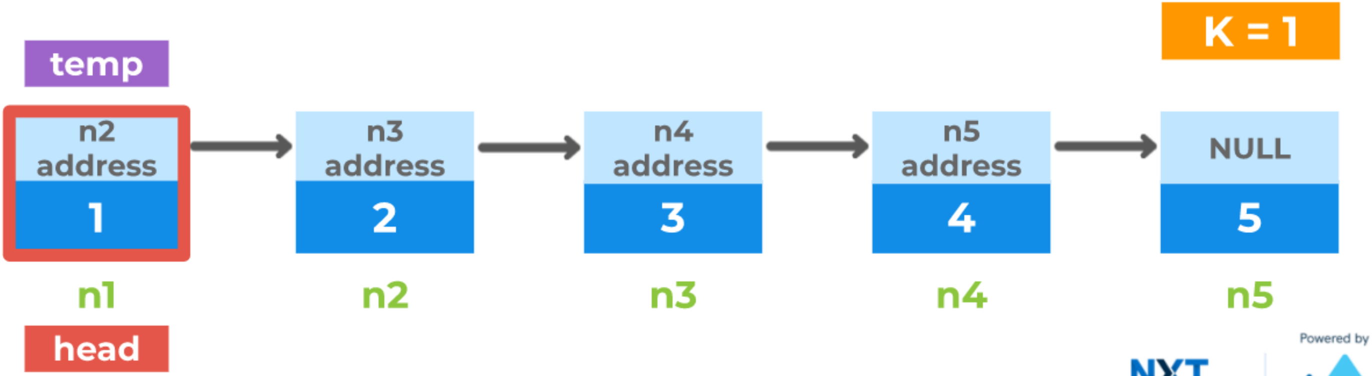
If the node to be deleted is the head node.



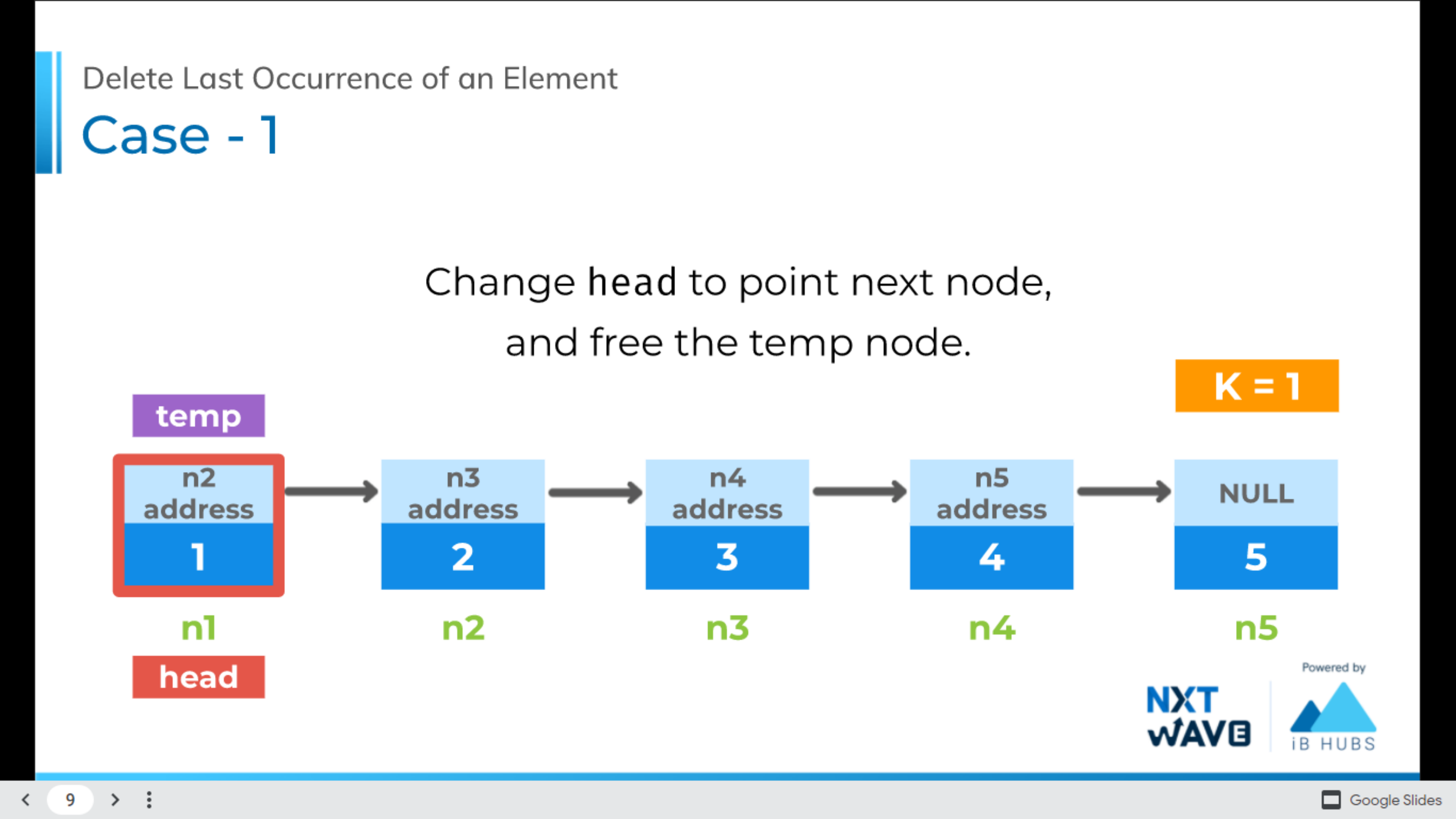
Delete Last Occurrence of an Element

# Case - 1

Create a new node `temp` pointing to the `head` node.



Powered by NXT WAVE iB HUBS



Delete Last Occurrence of an Element

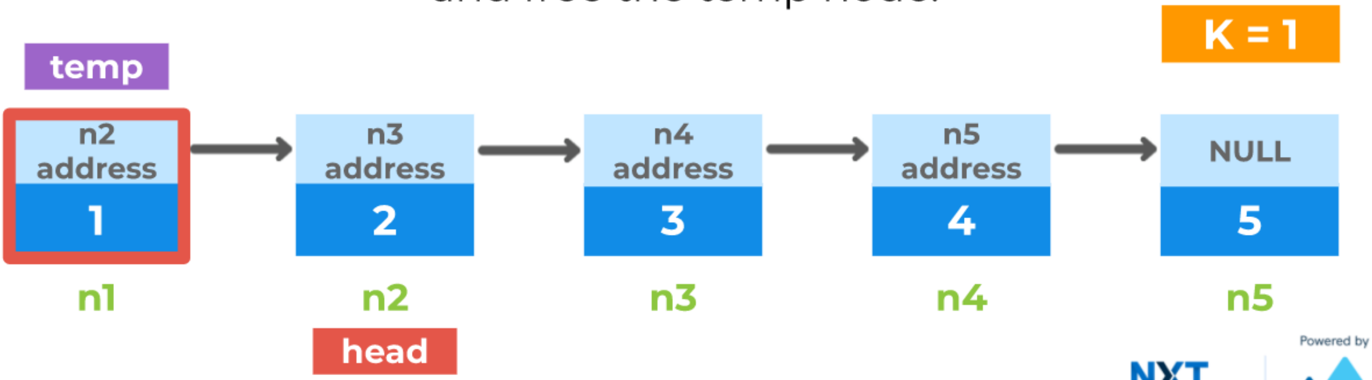
# Case - 1

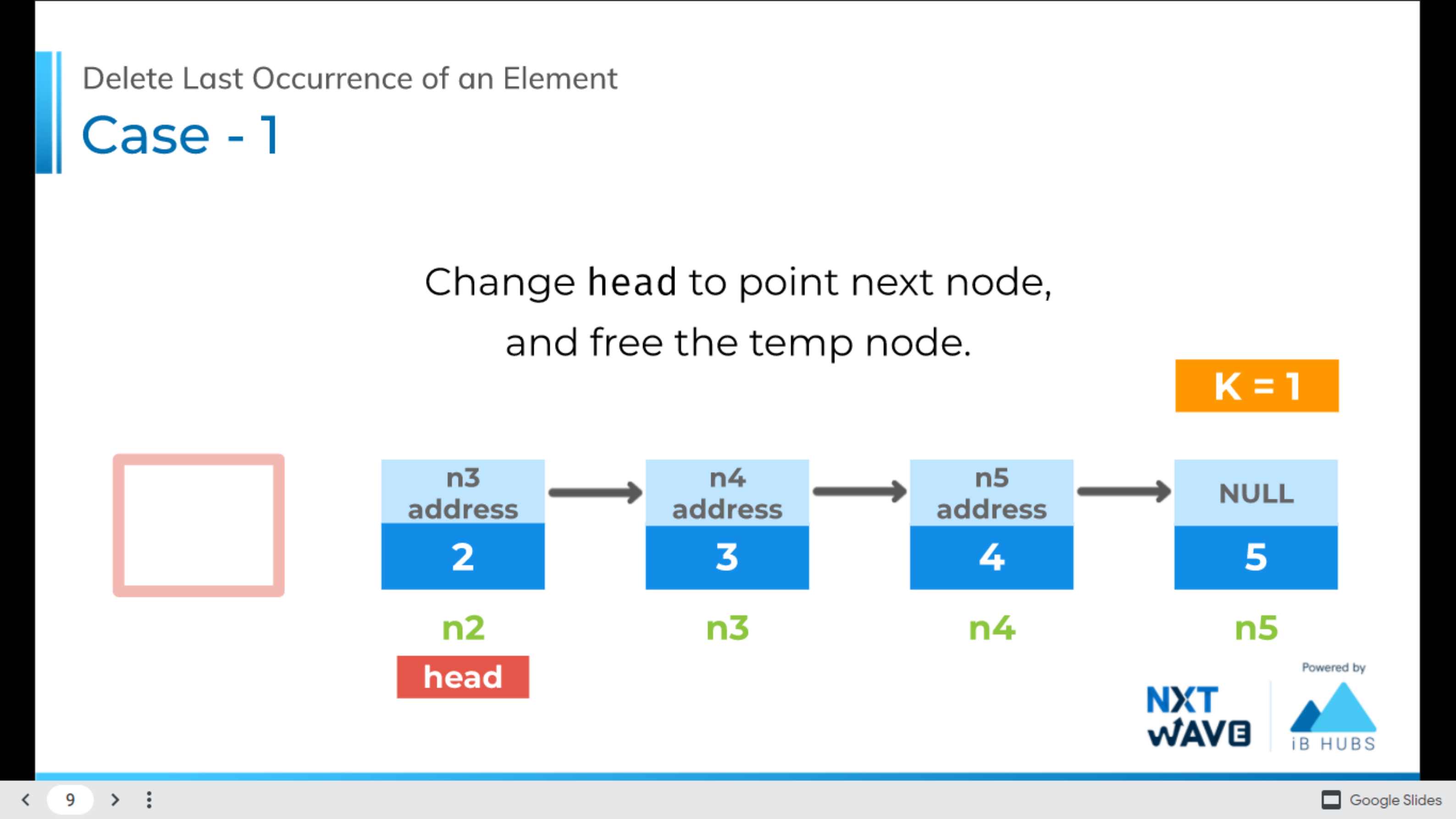
Change head to point next node,
and free the temp node.

Delete Last Occurrence of an Element

# Case - 1

Change head to point next node,
and free the temp node.

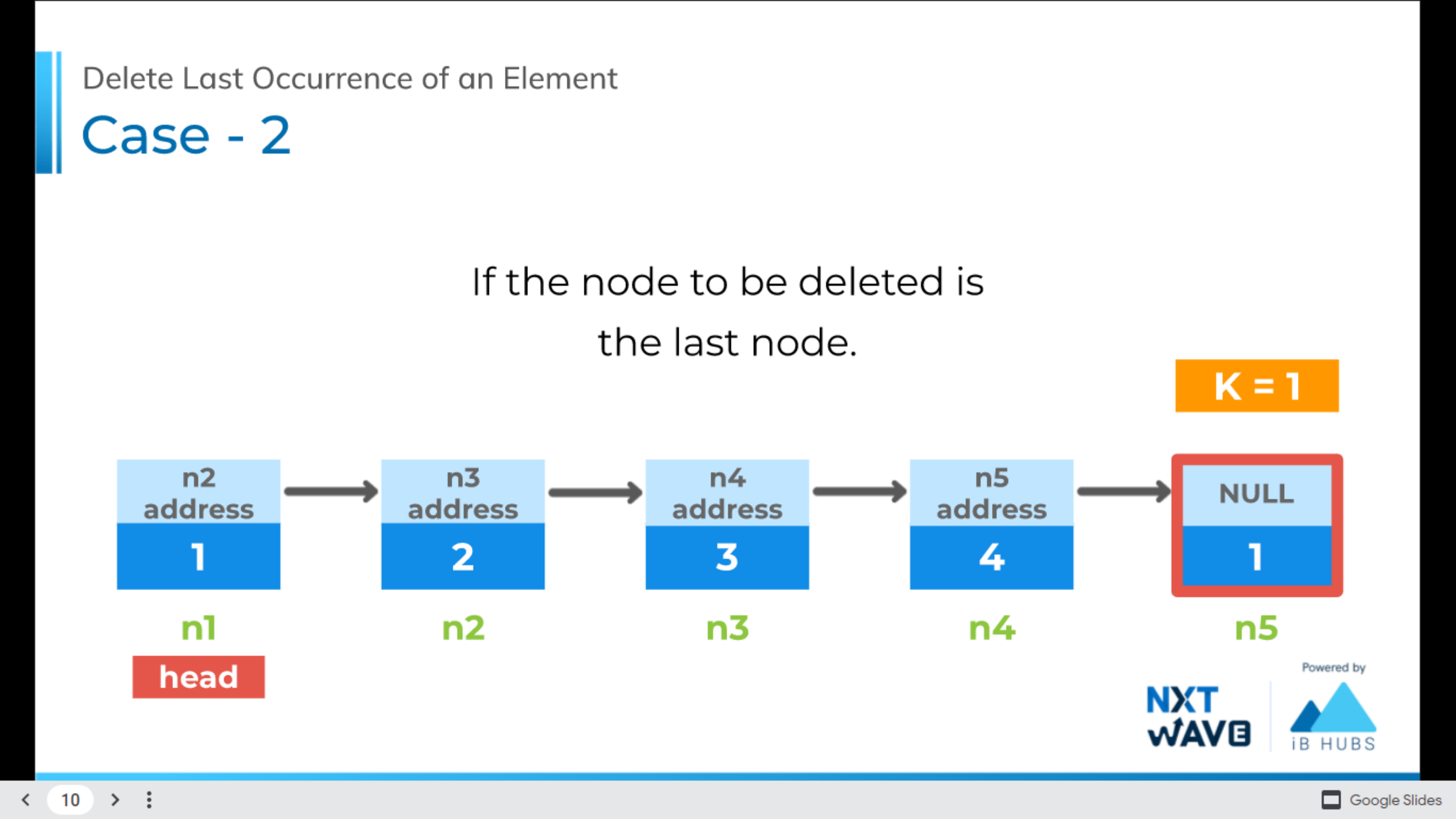




Delete Last Occurrence of an Element

## Case - 1

Change head to point next node,
and free the temp node.



Delete Last Occurrence of an Element

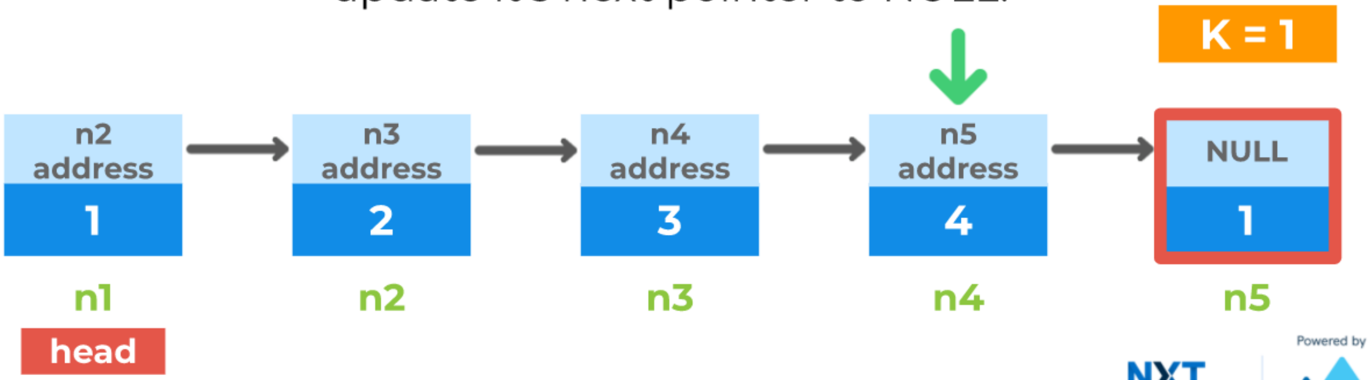
## Case - 2

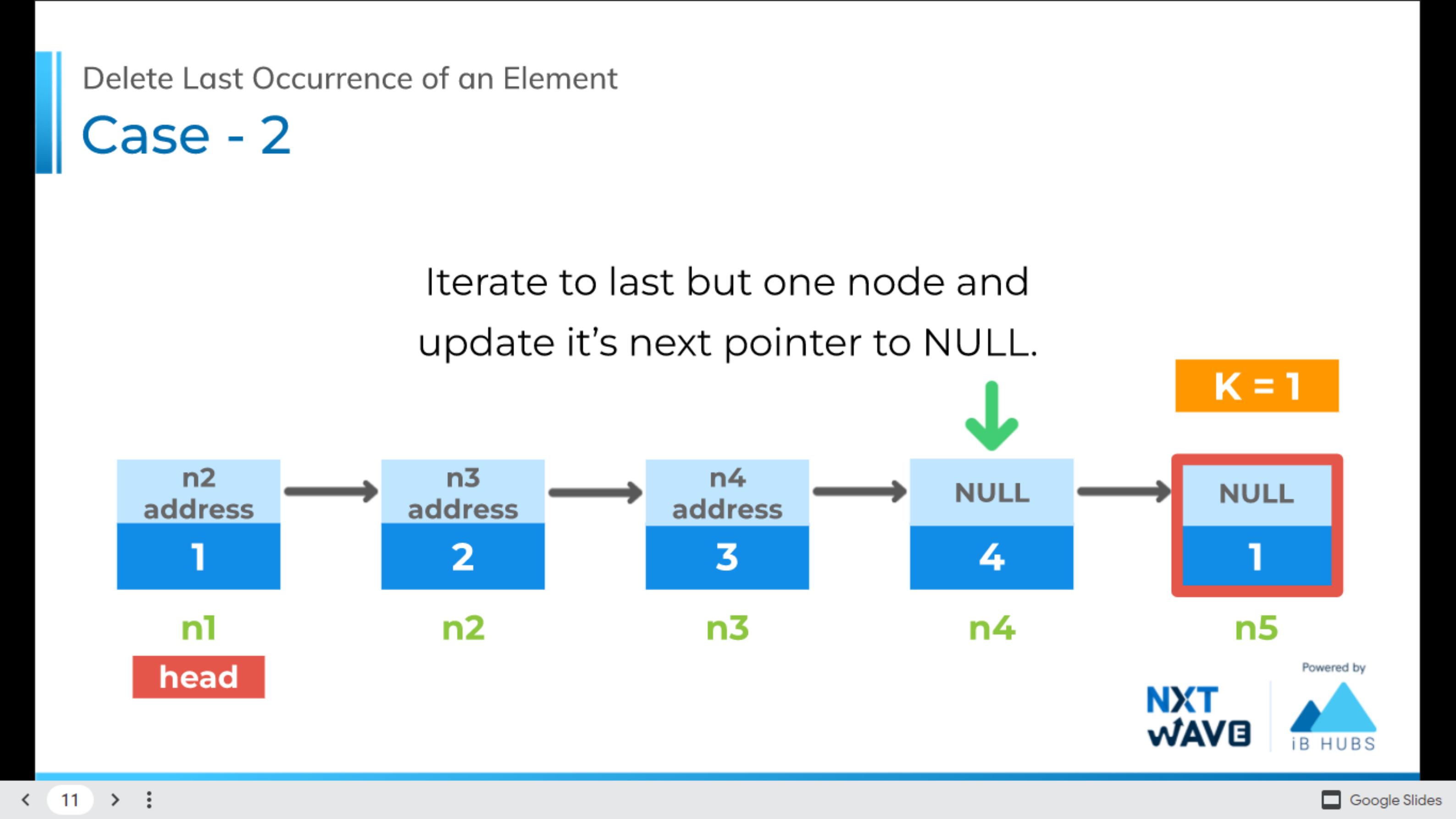
If the node to be deleted is the last node.

Delete Last Occurrence of an Element

# Case - 2

Iterate to last but one node and update it’s next pointer to NULL.





Delete Last Occurrence of an Element

## Case - 2

Iterate to last but one node and update it’s next pointer to NULL.

K = 1

n2 address | 1 → n3 address | 2 → n4 address | 3 → NULL | 4 → NULL | 1

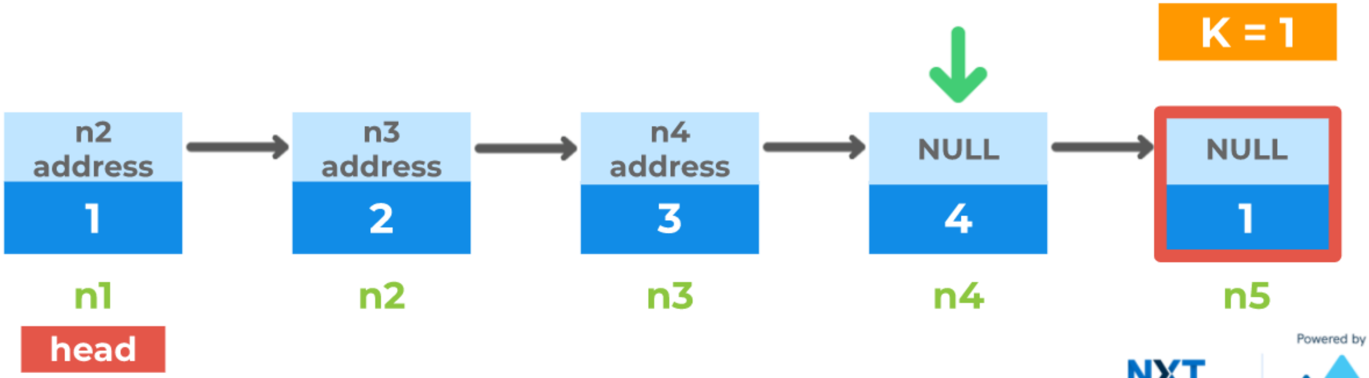
n1 n2 n3 n4 n5

head

Delete Last Occurrence of an Element

# Case - 2

Delete the last node.



K = 1

n2 address | 1 → n3 address | 2 → n4 address | 3 → NULL | 4 → NULL | 1

n1 n2 n3 n4 n5

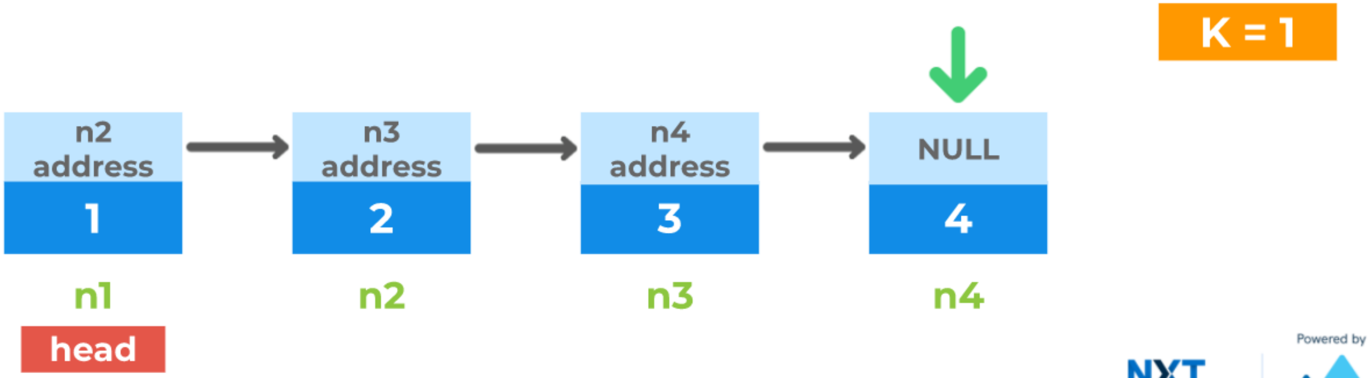
head

Powered by NXT WAVE iB HUBS

Delete Last Occurrence of an Element

# Case - 2

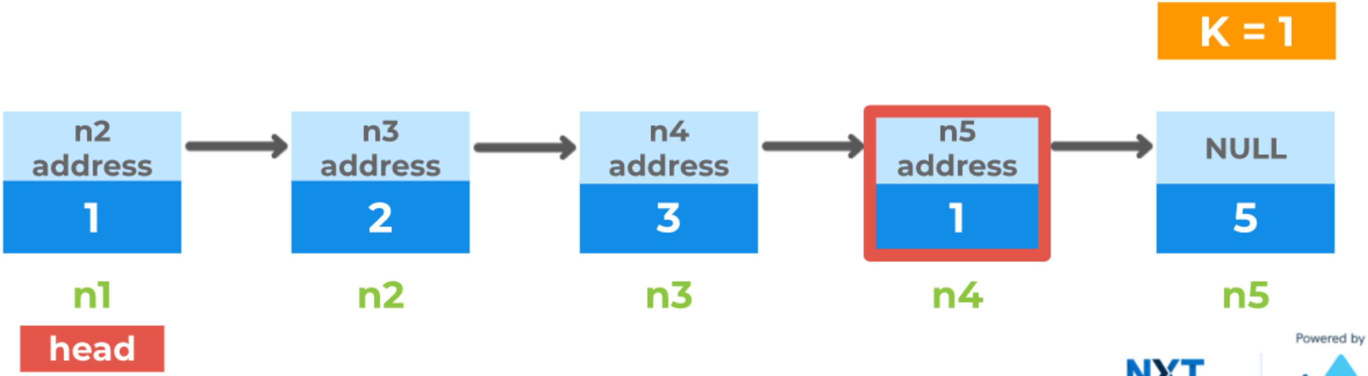
Delete the last node.



Delete Last Occurrence of an Element

# Case - 3

If the node to be deleted is somewhere inside the linked list.


K = 1

| n2 address | n3 address | n4 address | n5 address | NULL |
|---|---|---|---|---|
| 1 | 2 | 3 | 1 | 5 |
| n1 | n2 | n3 | n4 | n5 |

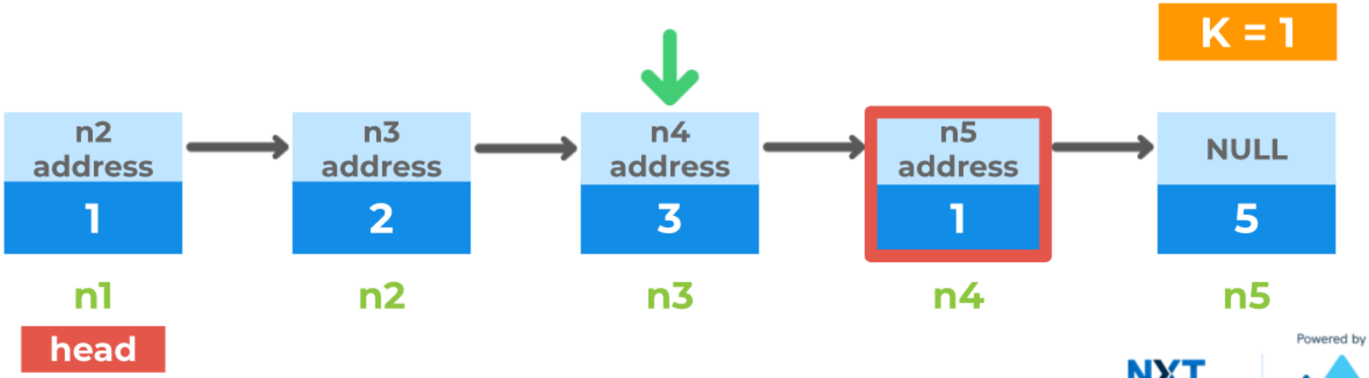
head

Powered by NXT WAVE iB HUBS

Delete Last Occurrence of an Element

## Case - 3

Iterate to the previous node of the node to be deleted.

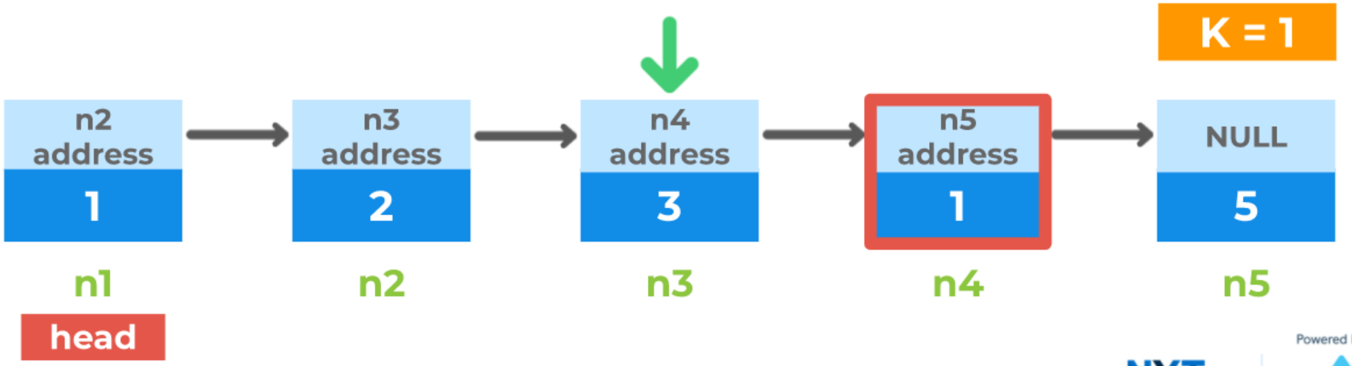


Powered by NXT WAVE iB HUBS

Delete Last Occurrence of an Element

# Case - 3

Update previous node next pointer to point the next node of the node to be deleted.



K = 1

n2 address
1
n1
head

n3 address
2
n2

n4 address
3
n3

n5 address
1
n4

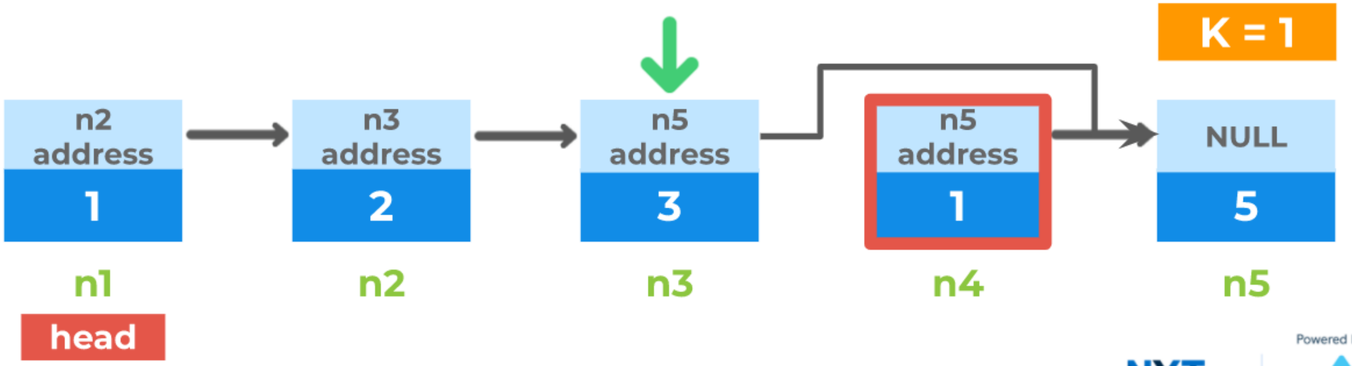
NULL
5
n5

Powered by NXT WAVE iB HUBS

Delete Last Occurrence of an Element

# Case - 3

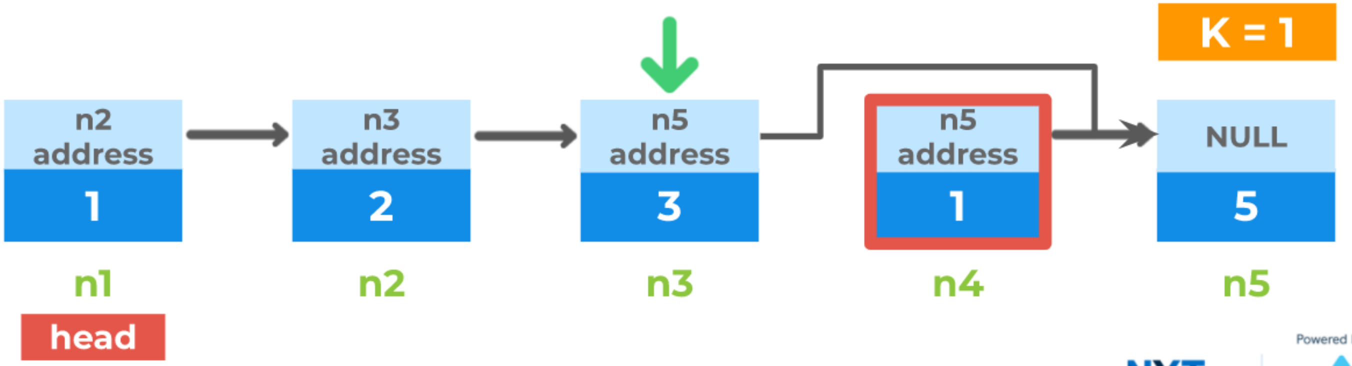
Update previous node next pointer to point the next node of the node to be deleted.



Delete Last Occurrence of an Element

# Case - 3

Delete node



K = 1

n2 address 1

n3 address 2

n5 address 3

n5 address 1

NULL 5

n1

n2

n3

n4

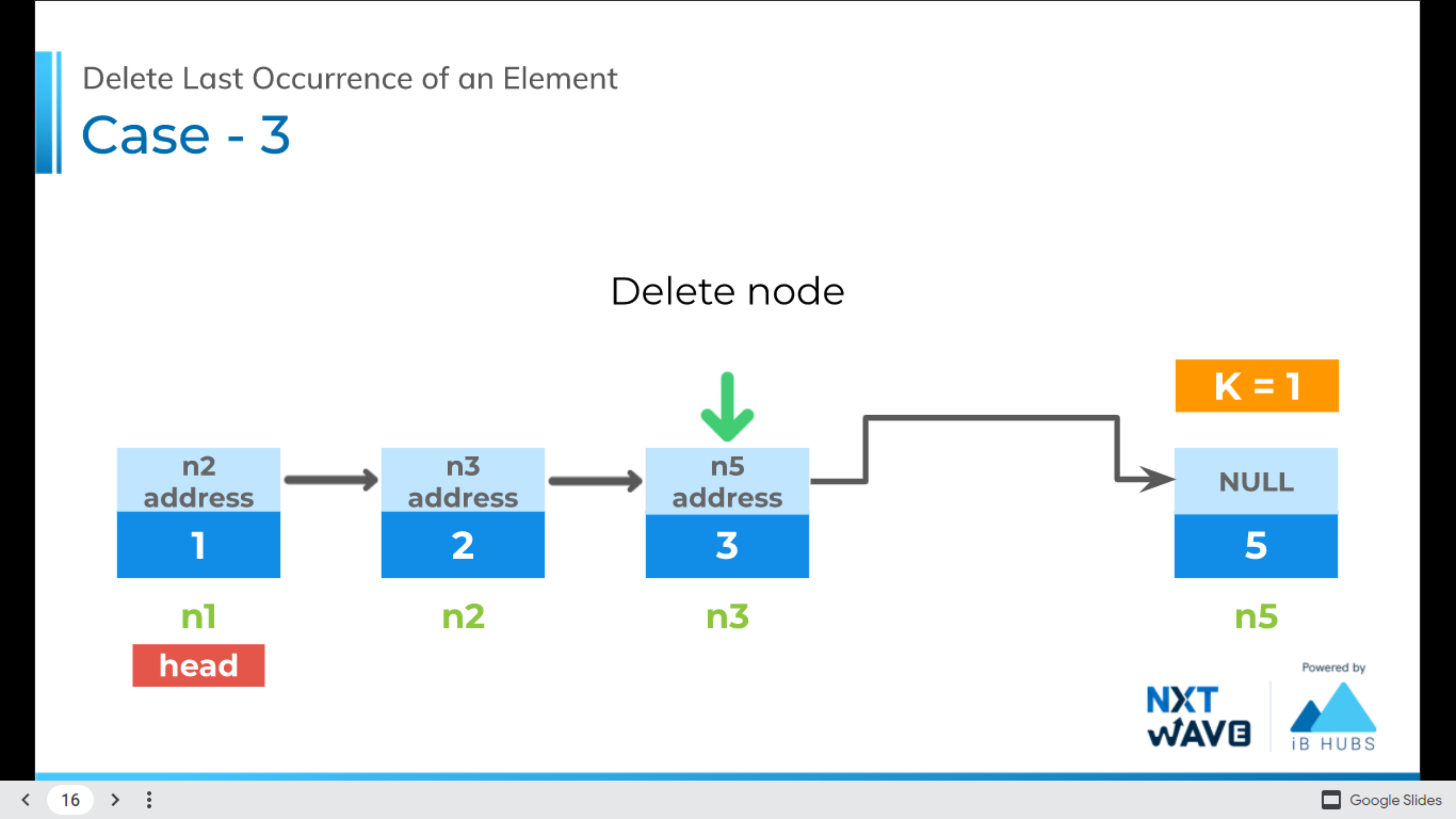
n5

head

Powered by

NXT WAVE

iB HUBS



Delete Last Occurrence of an Element

## Case - 3

Delete node

K = 1

n2 address | 1 — n1 — head

n3 address | 2 — n2

n5 address | 3 — n3

NULL | 5 — n5

# Key Takeaways

## Code Walkthrough

- ✓ Delete Last Occurrence of an Element

Powered by