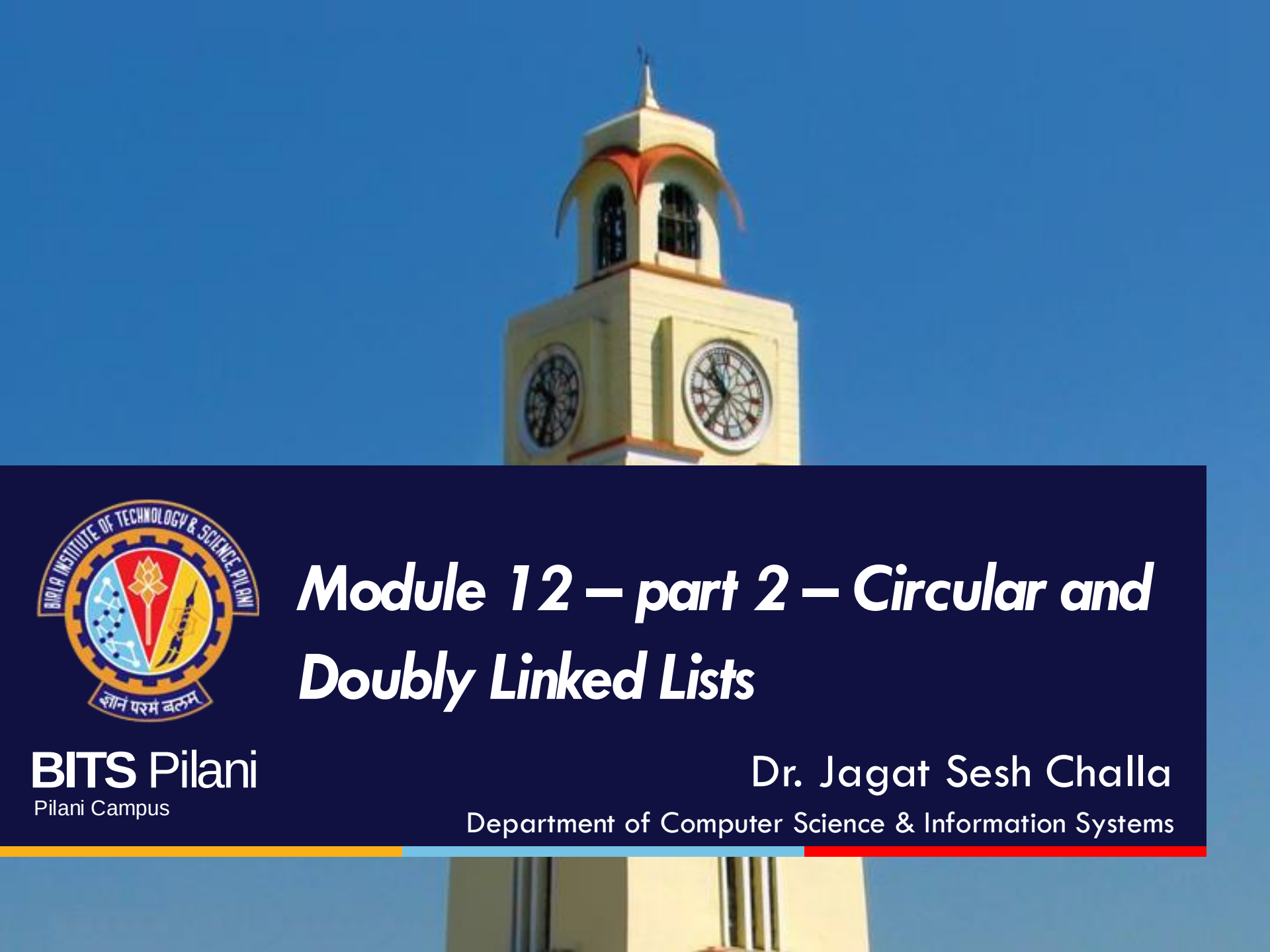


Module 12 – part 2 – Circular and Doubly Linked Lists

BITS Pilani
Pilani Campus

Dr. Jagat Sesh Challa
Department of Computer Science & Information Systems

Module Overview

innovate achieve lead

- Circular Linked Lists
- Doubly Linked Lists

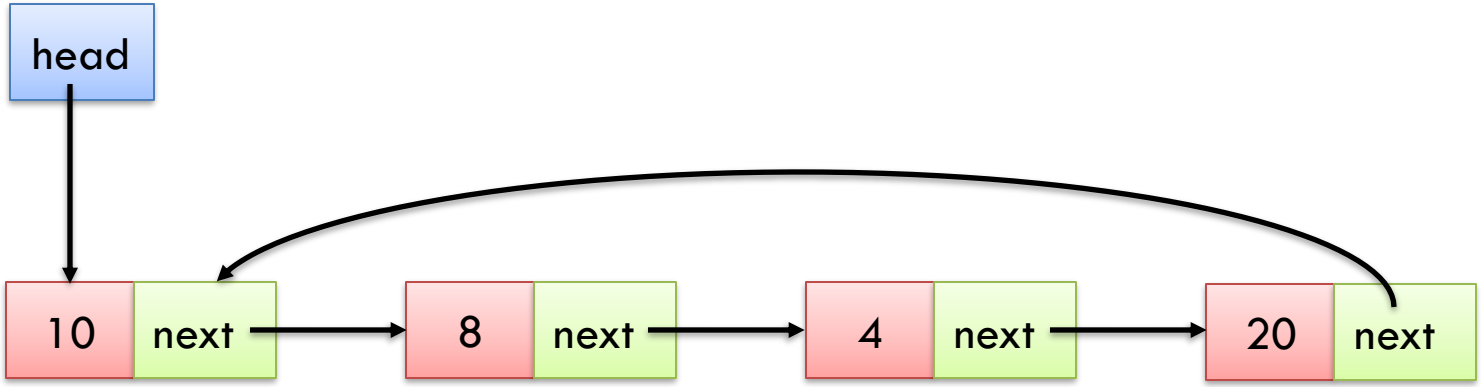
innovate achieve lead

BITS Pilani

Pilani Campus

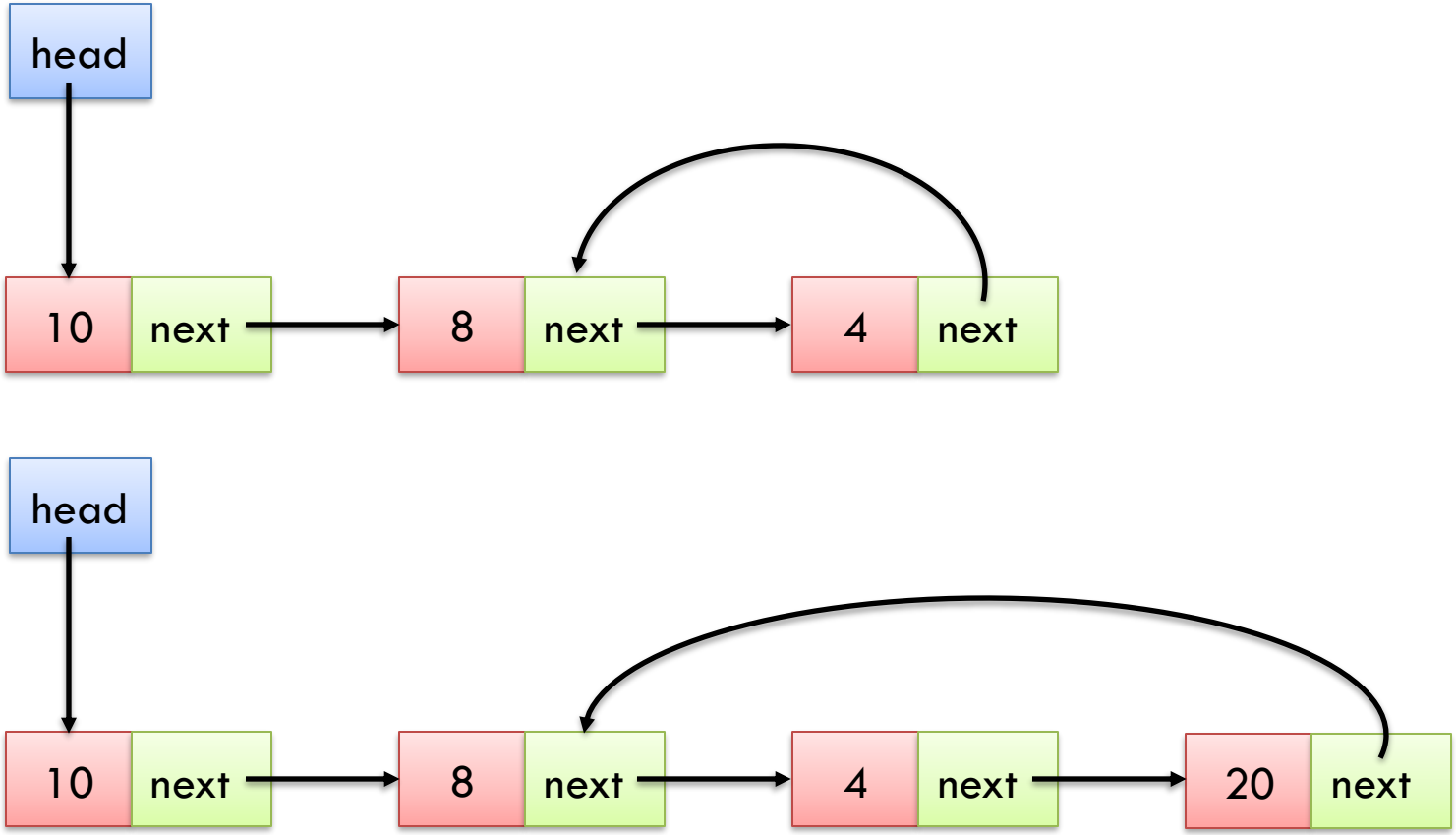
Circular Liked Lists and cycles in a Linked List

Circular Linked List



- The **next** of the last node *points to the first node*
- Result:
  - *Traversing the list is an infinite operation as you will never encounter a NULL pointer*

More examples



Detecting cycles in linked list

Cycle in a linked list

innovate achieve lead

Solution 1:

- Traverse the list. While traversing, store the addresses of all the visited nodes in an array/table.
  - When we traverse from node1 to node2, check if the address of node2 already exists in the table. If **YES**, a cycle is detected. If not, add address of node 2 to the table and go forward.
  - If you encounter a NULL in the traversal, then the list doesn't have a cycle.

Cycle in a linked list

Solution 2: This solution requires modifications to the basic linked list data structure.

- Have a **visited flag** with each node
- Traverse the linked list and keep marking visited nodes
- If you see a visited node again then there is a cycle
- If you encounter a NULL in the traversal, then the list doesn't have a cycle.

Cycle in a linked list

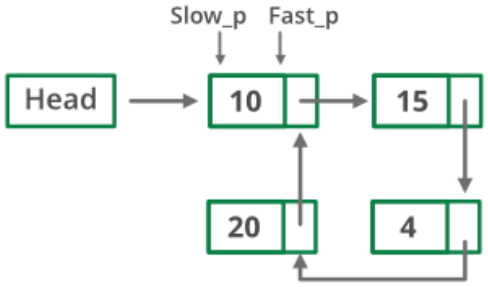
innovate achieve lead

Solution 3: This is the fastest method

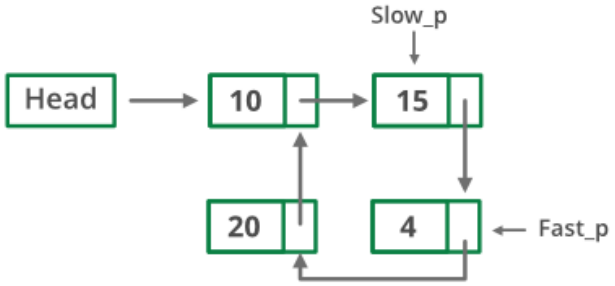
- Traverse linked list using two pointers (**slow_p** & **fast_p**)
- Move (slow_p) by one node and fast_p by two.
- If these pointers meet at the same node then there is a cycle.
- If pointers do not meet then linked list doesn't have a cycle. OR if either pointer encounters a NULL in the traversal, then the list doesn't have a cycle.

Solution 3 illustrated

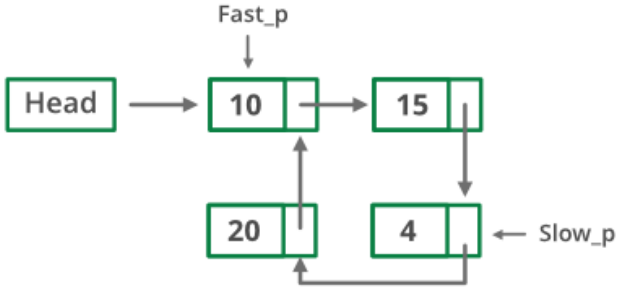
Initially :



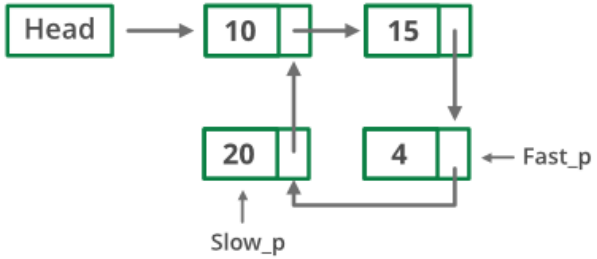
Step 1:



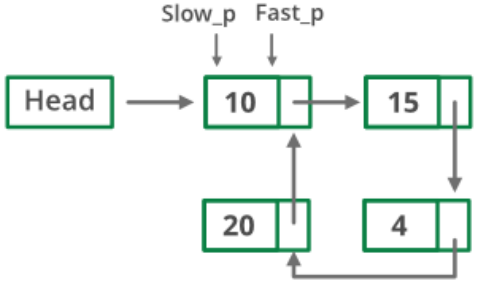
Step 2:



Step 3:



Step 4:



Loop Detected

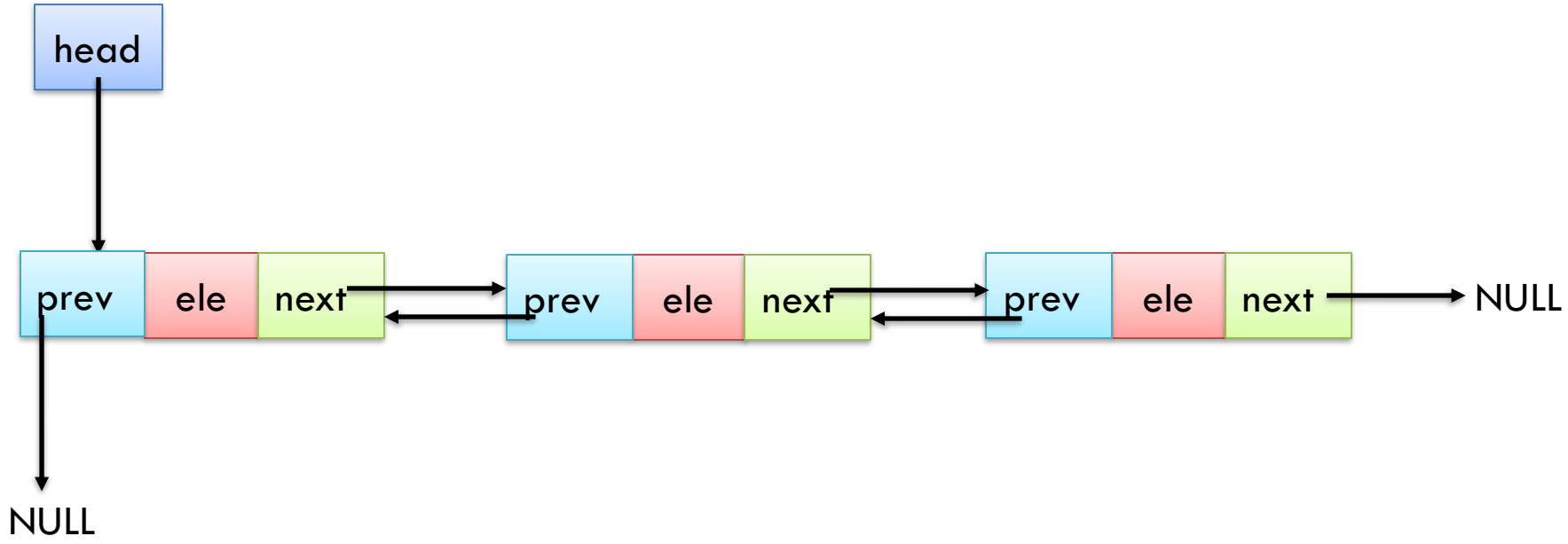
innovate achieve lead

BITS Pilani

Pilani Campus

Doubly Liked Lists

Doubly Linked Lists



- Each node now stores:
  - *value of the element stored at that node:* **ele**
  - *address of the next node:* **next**
  - *address of the previous node:* **prev**
- *The* **head node** *contains the* ***address of the first node***
- *The* ***next of the last node points*** *to* **NULL**, *meaning end of the list*
- *The* ***prev of the first node points*** *to* **NULL**, *meaning beginning of the list*

Supports two-way traversal of linked lists

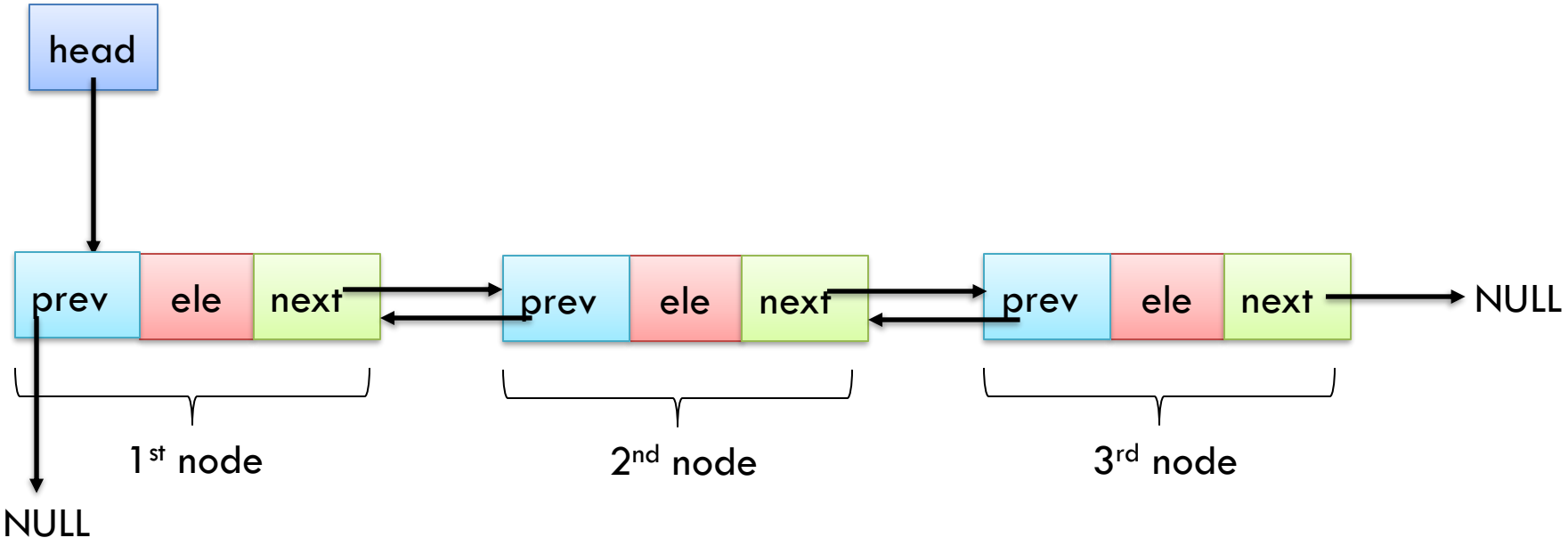
innovate achieve lead

BITS Pilani

Pilani Campus

Doubly Liked Lists – Implementation

Structure definitions for doubly linked lists



Consider that our doubly linked list stores integer elements.

```
struct dllnode{
   int ele;
   struct dllnode * next;
   struct dllnode * prev;
};
```

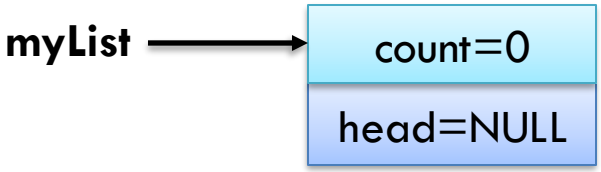
```
struct doubly_linked_list{
   int count;
   struct dllnode * head;
};
```

Creating a new doubly linked list


```
typedef struct dllnode * DLLNODE;
struct dllnode{
   int ele;
   DLLNODE next;
   DLLNODE prev;
};
```

```
typdef struct doubly_linked_list * DLIST;
struct doubly_linked_list{
   int count;
   DLLNODE head;
};
```

```
DLIST createNewList(){
   DLIST myList;
   myList = (DLIST) malloc(sizeof(struct doubly_linked_list));
   // myList = (DLIST) malloc(sizeof(*myList));
   myList->count=0;
   myList->head=NULL;
   return myList;
}
```

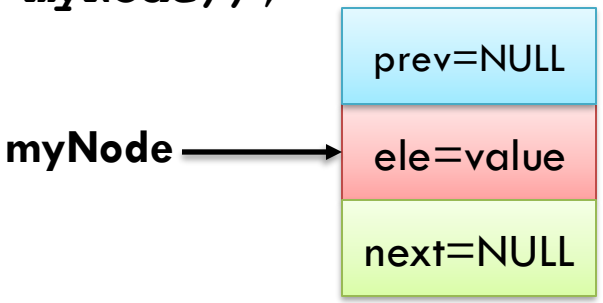


Creating a new node


```
typedef struct dllnode * DLLNODE;
struct dllnode{
   int ele;
   DLLNODE next;
   DLLNODE prev;
};
```

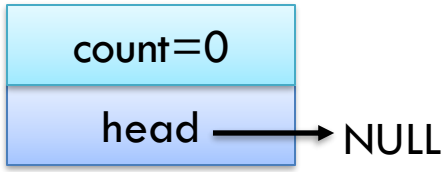
```
typdef struct doubly_linked_list * DLIST;
struct doubly_linked_list{
   int count;
   DLLNODE head;
};
```

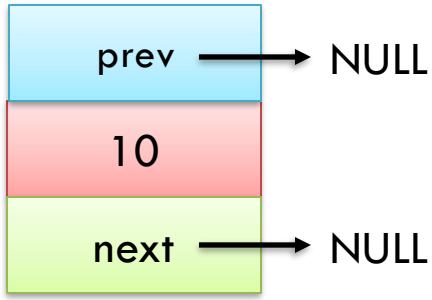
```
DLLNODE createNewNode(int value){
   DLLNODE myNode;
   myNode = (DLLNODE) malloc(sizeof(struct dllnode));
   // myList = (DLLNODE) malloc(sizeof(*myNode));
   myNode->ele=value;
   myNode->next=NULL;
   myNode->prev=NULL;
   return myNode;
}
```

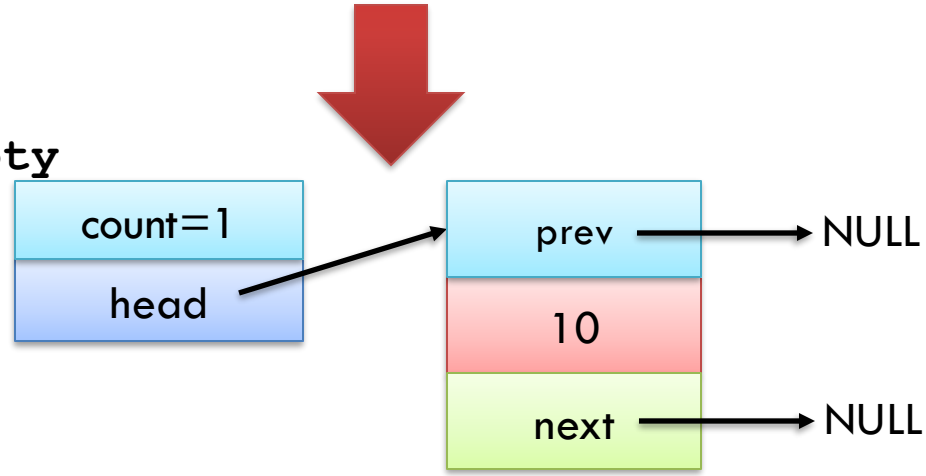


Inserting a node into the list

```
void insertNodeIntoList(DLLNODE n1, DLIST l1){
   // case when list is empty
   if(l1->count == 0) {
        l1->head = n1;
        n1->next = NULL;
        n1->prev = NULL;
        l1->count++;
   }
   // case when list is non empty
   else {
        ... ...
   }
}
```

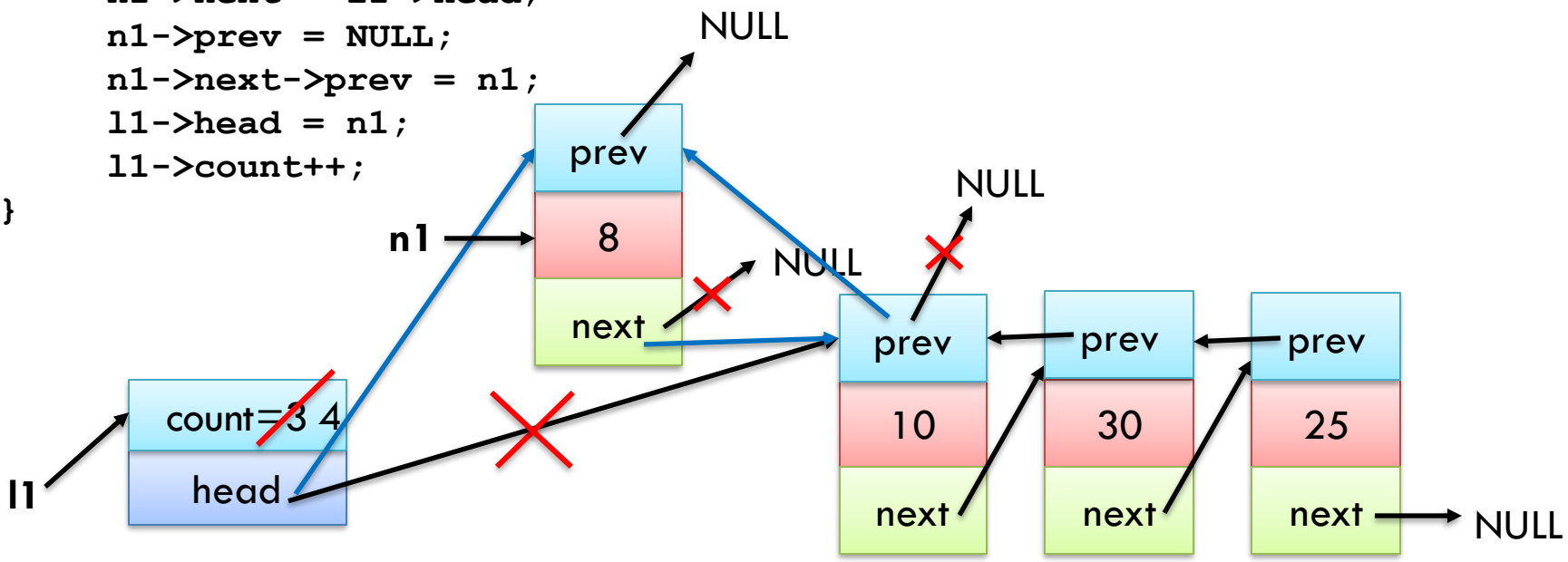






Inserting a node into the list (contd.)

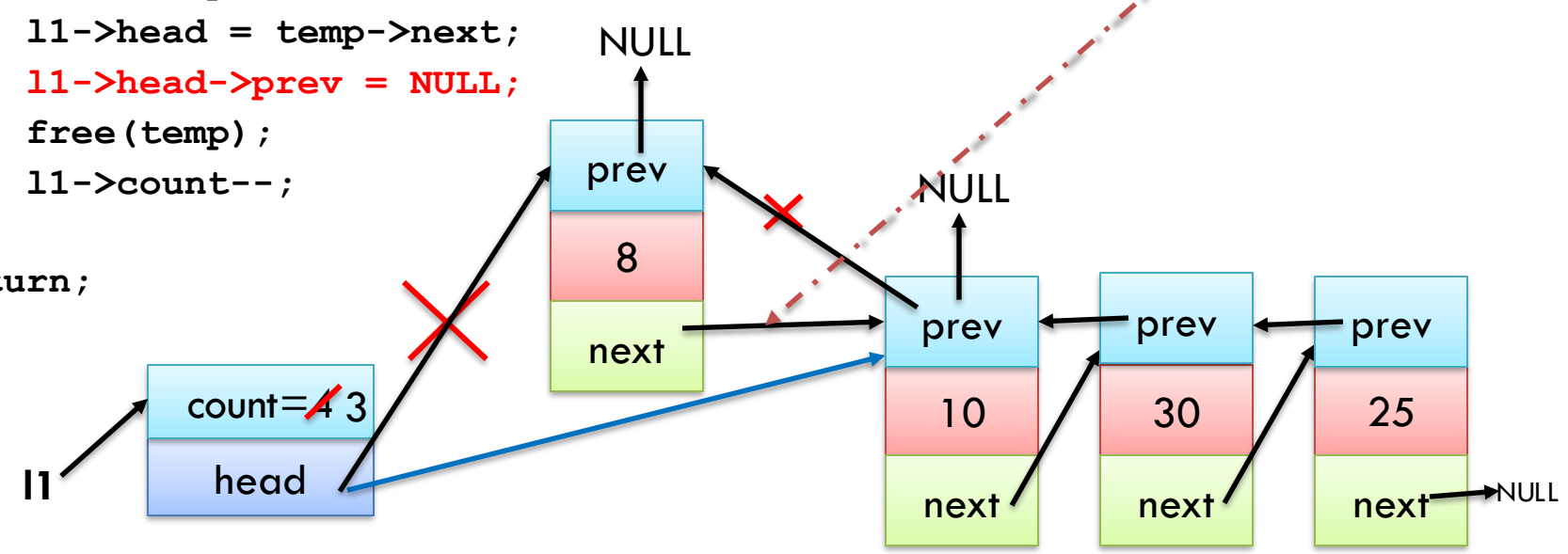
```
void insertNodeIntoList(DLLNODE n1, DLIST l1){
    // case when list is empty
    if(l1->count == 0) {
          ... ...
    }
    // case when list is non empty
    else {
          n1->next = l1->head;
          n1->prev = NULL;
          n1->next->prev = n1;
          l1->head = n1;
          l1->count++;
    }
}
```



Removing a node from the beginning of the list

```
void removeFirstNode(LIST l1)
{
    if (l1->count == 0)
    {
        printf("List is empty. Nothing to remove\n");
    }
    else
    {
        NODE temp = l1->head;
        l1->head = temp->next;
        l1->head->prev = NULL;
        free(temp);
        l1->count--;
    }
    return;
}
```

Do we have to bother about removing this link?



Other functions (exercise)

Exercise: Implement the following functions for a linked list:

- `insertNodeAtEnd(DLIST mylist, DLLNODE n1)` : inserts n1 at the end of mylist
- `insertAfter(DLIST mylist, DLLNODE n1, int v)`: inserts n1 into mylist after a node containing a value v
- `removeLastNode(DLIST mylist)` : removes the last node from mylist
- `search(int data, DLIST mylist)`: returns the node that contains its ele=data
- `printList_forward(DLIST mylist)`: prints the elements present in the entire list in a sequential fashion in forward direction starting from the first element
- `printList_backward(DLIST mylist)`: prints the elements present in the entire list in a sequential fashion in backward direction starting from the last element
- `removeElement(int data, DLIST mylist)`: removes the node that has its ele=data
- `isEmpty(DLIST mylist)`: checks if the list is empty or not
- Modify the insert/delete functions to first check whether the list is empty using `isEmpty()` function.

innovate achieve lead

BITS Pilani

Pilani Campus

Thank you

Q & A