CS & IT ENGINEERING

Data structures & Programming

Linked List

Lec- 07

By- Pankaj Sharma sir

Linked List -7

Q. 1

Let SLLdel be a function that deletes a node in a singly-linked list given a pointer to the node and a pointer to the head of the list. Similarly, let DLLdel be another function that deletes a node in a doubly-linked list given a pointer to the node and a pointer to the head of the list.
Let n denote the number of nodes in each of the linked lists.
Which one of the following choices is TRUE about the worst-case time complexity of SLLdel and DLLdel?

[GATE-2023-CS:1M]

A SLLdel is O(1) and DLLdel is O(n)

B Both SLLdel and DLLdel are O(log(n))

C Both SLLdel and DLLdel are O(1)

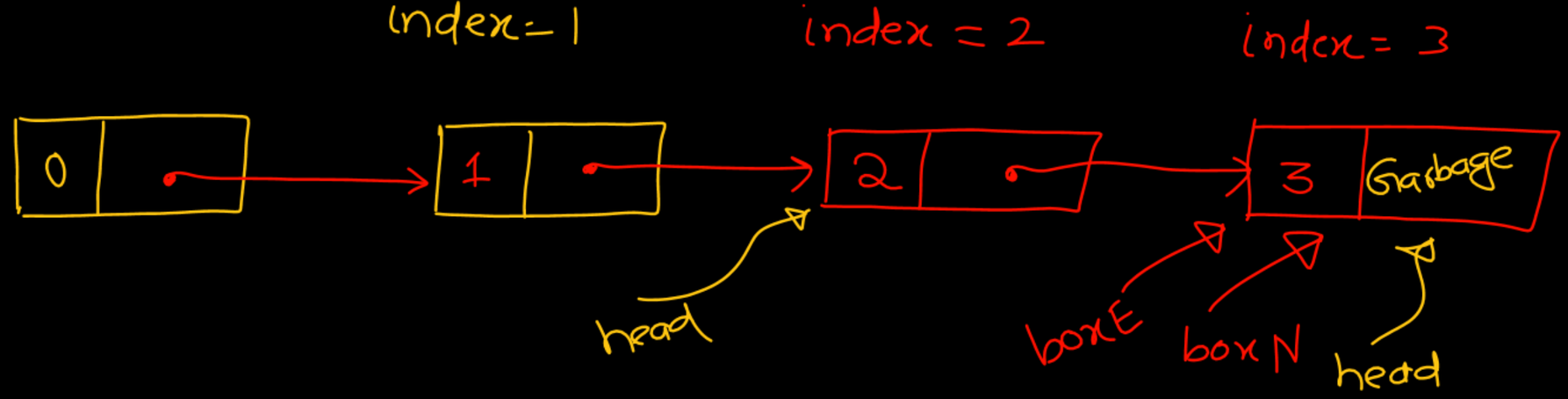
D SLLdel is O(n) and DLLdel is O(1)

Q. 3

Consider the following ANSI C program: [GATE-2021-CS:2M]

```
#include <stdio.h>
#include <stdlib.h>
struct Node{
        int value;
        struct Node *next;};
int main( ){
    struct Node *boxE, *head, *boxN; int index = 0;
    boxE = head = (struct Node *) malloc(size of(struct Node));
    head → value = index;
    for (index = 1; index <= 3; index++){
        boxN = (struct Node *) malloc(size of(struct Node));
        boxE → next = boxN;
        boxN → value = index;
        boxE = boxN; }
```

template



index=0

value at index 0 is 0

head point to 2nd node

value at index 1 is 1

index=1

value at index 1 is 1

head point to 3rd node

value at index 2 is 2

index=2

value at index 2 is 2

head point to last node

value at index 3 is 3

index=3

value at index 3 is 3

Read ⇒ Garbage address

bf ⇒ may produce error

```
for (index = 0; index ≤ 3; index++) {
        printf ("Value at index %d is %d\n", index, head → value);
        head = head → next;
        printf ("Value at index %d is %d\n", index+1, head → value); } }
```

Which one of the statements below is correct about the program?

A. Upon execution, the program creates a linked-list of five nodes.

B. Upon execution, the program goes into an infinite loop.

C. It has a missing return which will be reported as an error by the compiler.

D. It dereferences an uninitialized pointer that may result in a run-time error.

Q. 4

Consider the C code fragment given below.

[GATE-2017-CS:1M]

```
typedef struct node{
    int data;
    node* next;
 } node;
void join (node *m, node *n){
    node *p = n;
    while (p → next ! = NULL){
    p = p → next;
   }
   p → next = m;
}
```

Assuming that m and n point to valid NULL-terminated linked lists, invocation of join will

A append list m to the end of list n for all inputs.

B either cause a null pointer dereference or append list m to the end of list n.

C cause a null pointer dereference for all inputs.

D append list n to the end of list m for all inputs.

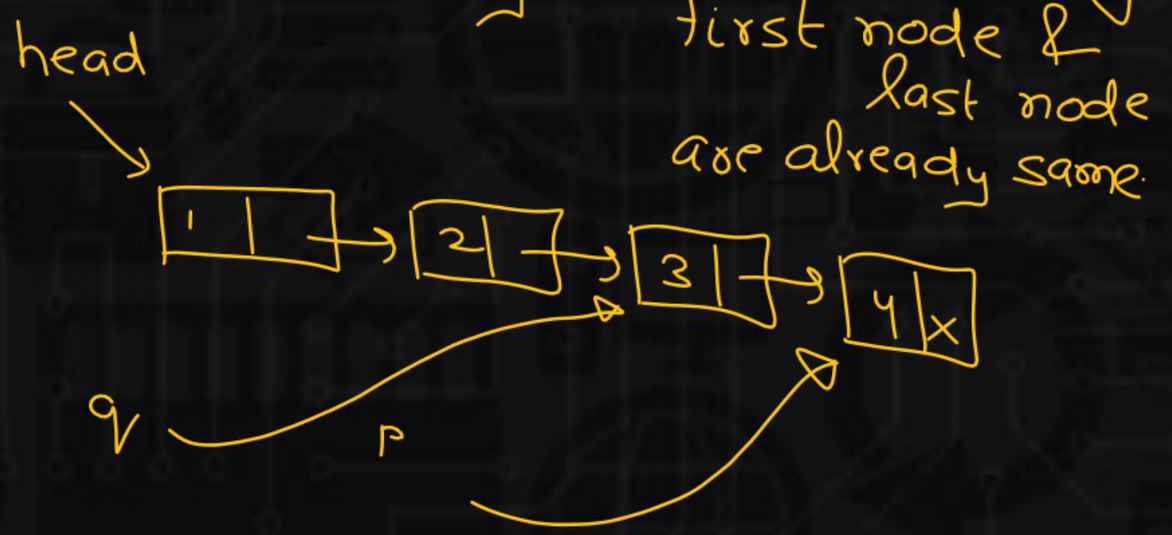
Q. 5

The following C function takes a singly-linked list as input argument. It modified the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank.

[GATE-2010-CS:2M]

```
typedef struct node{
        int value;
        struct node * next;
 }Node;
Node * move_ to _ front (Node * head){
        Node * p, *q;
        if ((head = = NULL)|| (head→next = =NULL))
        return head;
q = NULL;
p = head;
        while (p →next ! = NULL){
         q = p;
         p = p → next;
          }

_____
 return head;
}
```

0 or 1 node
→ do nothing
first node & last node are already same.



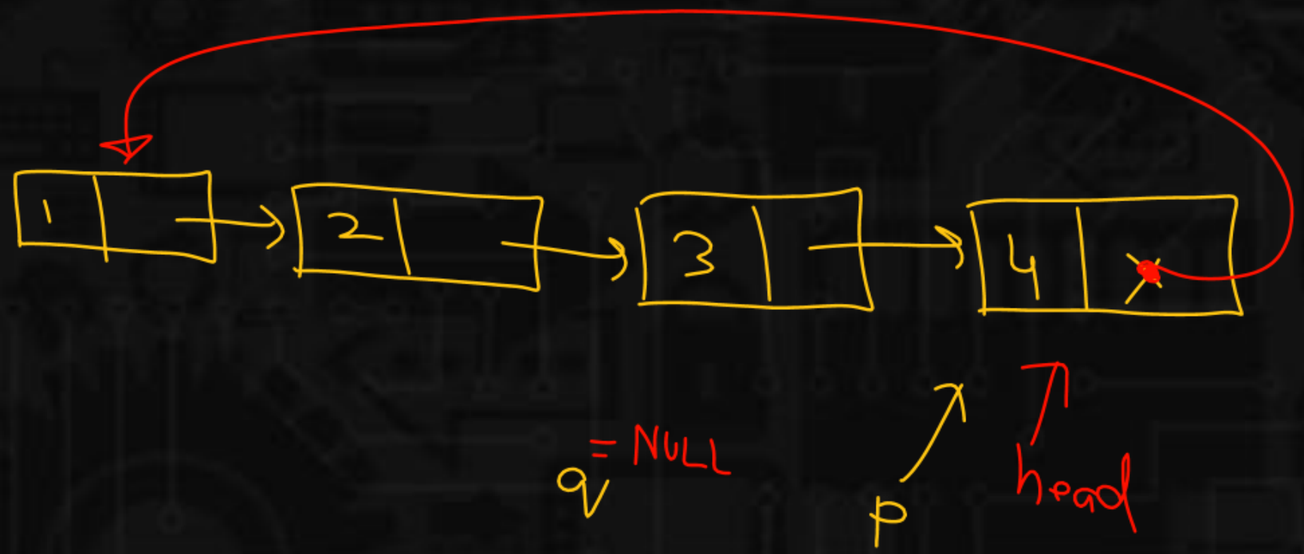
Choose the correct alternative to replace the blank line.

A $q = NULL ; p \rightarrow next = head ; head = p;$

B $q \rightarrow next = NULL; head = p; p \rightarrow next = head;$

C $head = p; p \rightarrow next = q; q \rightarrow next = NULL;$

D $q \rightarrow next = NULL; p \rightarrow next = head; head = p;$

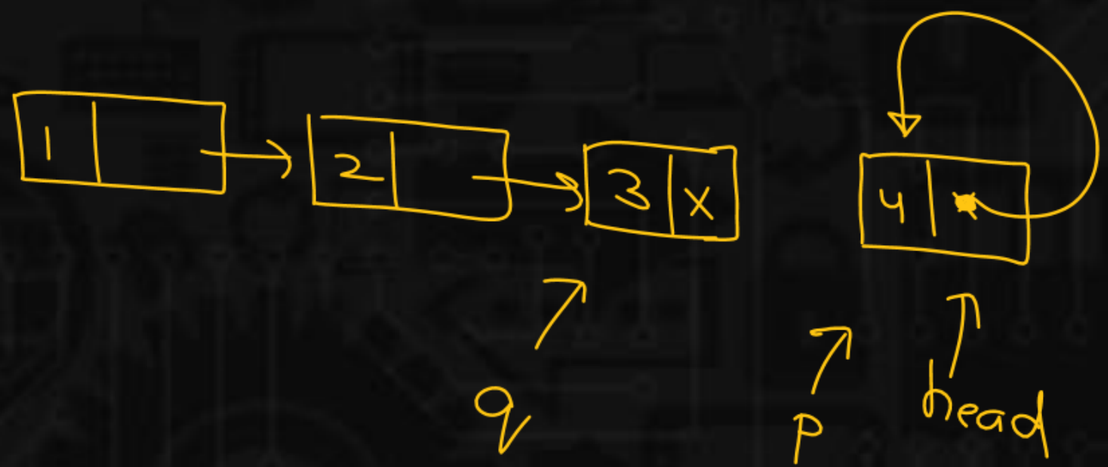


A q = NULL ; p → next = head ; head = p;

B q→ next = NULL; head = p; p → next = head;

C head = p; p → next = q; q→ next = NULL;

D q→ next = NULL; p→ next = head; head = p;

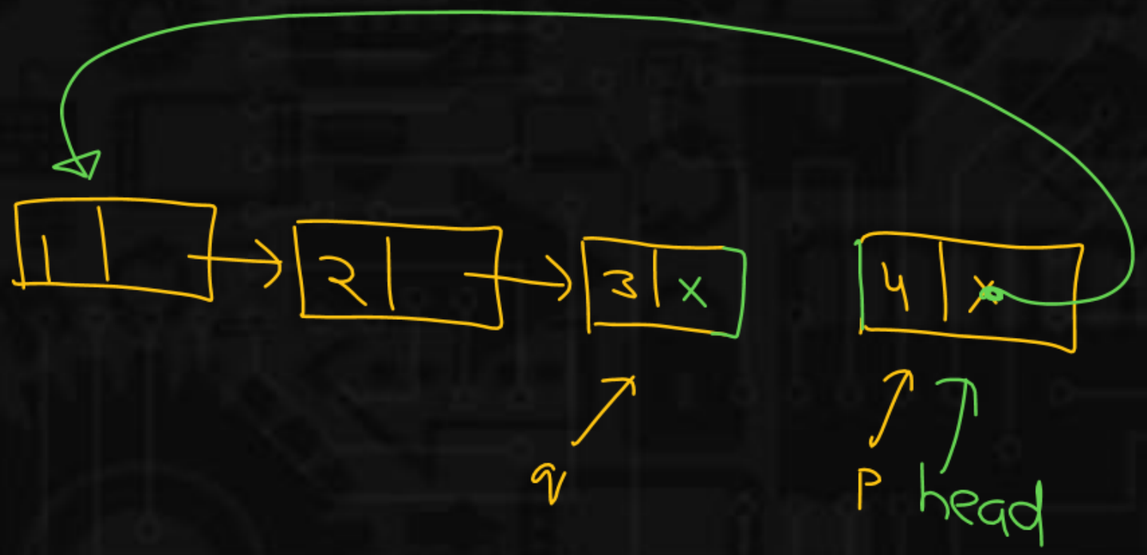


A $q = NULL ; p \rightarrow next = head ; head = p;$

B $q \rightarrow next = NULL; head = p; p \rightarrow next = head;$

C $head = p; p \rightarrow next = q; q \rightarrow next = NULL;$

D $q \rightarrow next = NULL; p \rightarrow next = head; head = p;$



Q. 6

The following C function takes a singly-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

[GATE-2008-CS:2M]

```
struct node{
        int value;
        struct node * next;
};
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:
                   }
}
```

A 1, 2, 3, 4, 5, 6, 7

B 2, 1, 4, 3, 6, 5, 7

C 1, 3, 2, 5, 4, 7, 6

D 2, 3, 4, 5, 6, 7, 1

Types of Linked List

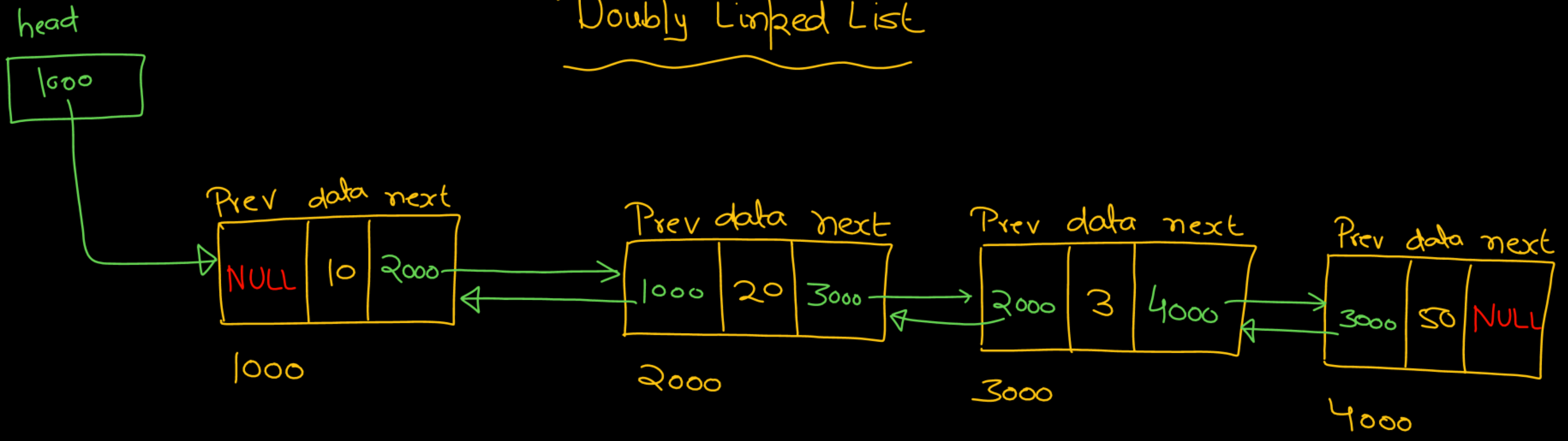
1) Singly Linked list ✓ done

2) Doubly L.L.

3) Circular L.L

4) Header L.L

Doubly Linked List

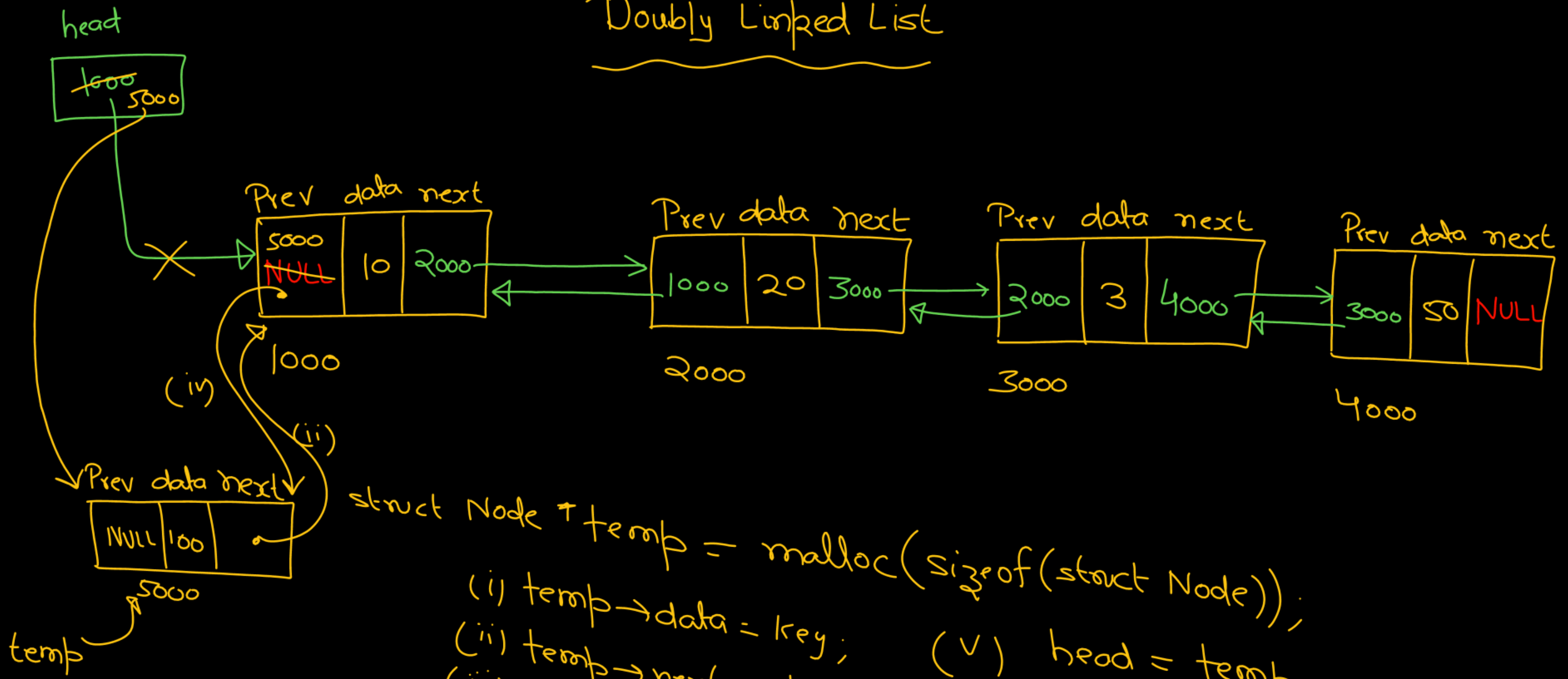



```
struct Node {
    struct Node *prev;
    int data;
    struct Node *next;
}
```

Traversal → same as SLL

Insertion

Doubly Linked List




```
struct Node *temp = malloc(sizeof(struct Node));
```

(i) temp→data = key;

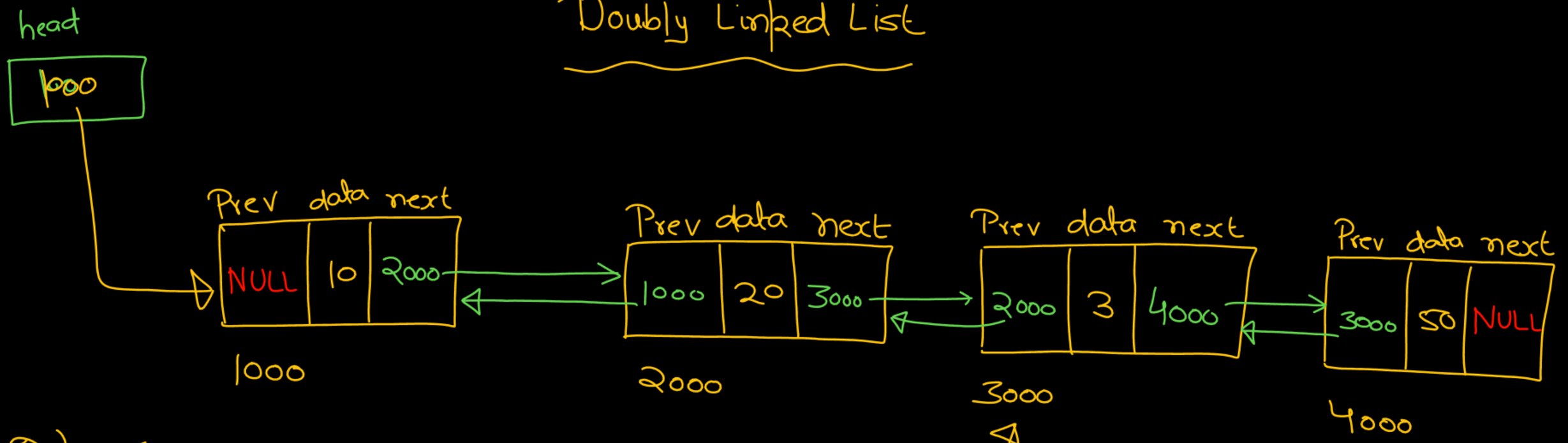
(ii) temp→next = head

(iii) temp→Prev = NULL

(iv) head→Prev = temp

(v) head = temp

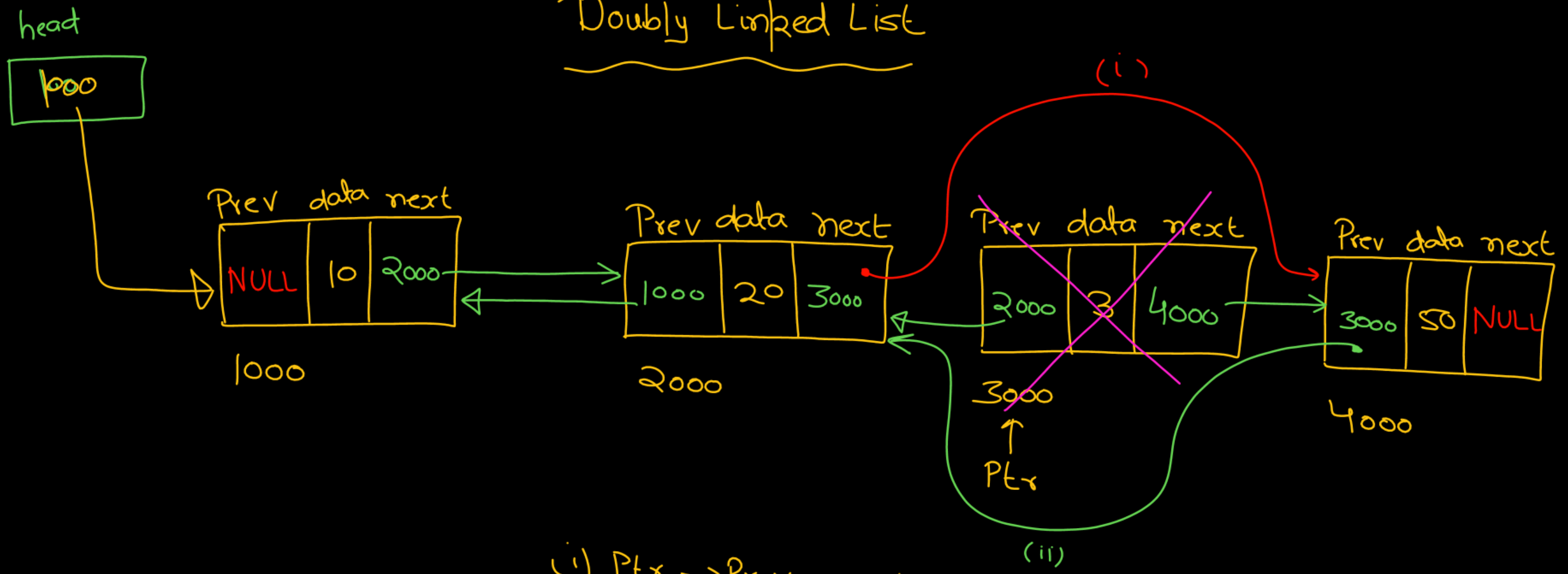
Doubly Linked List



8) Given a pointer to a node, delete that node.

Ptr

Doubly Linked List

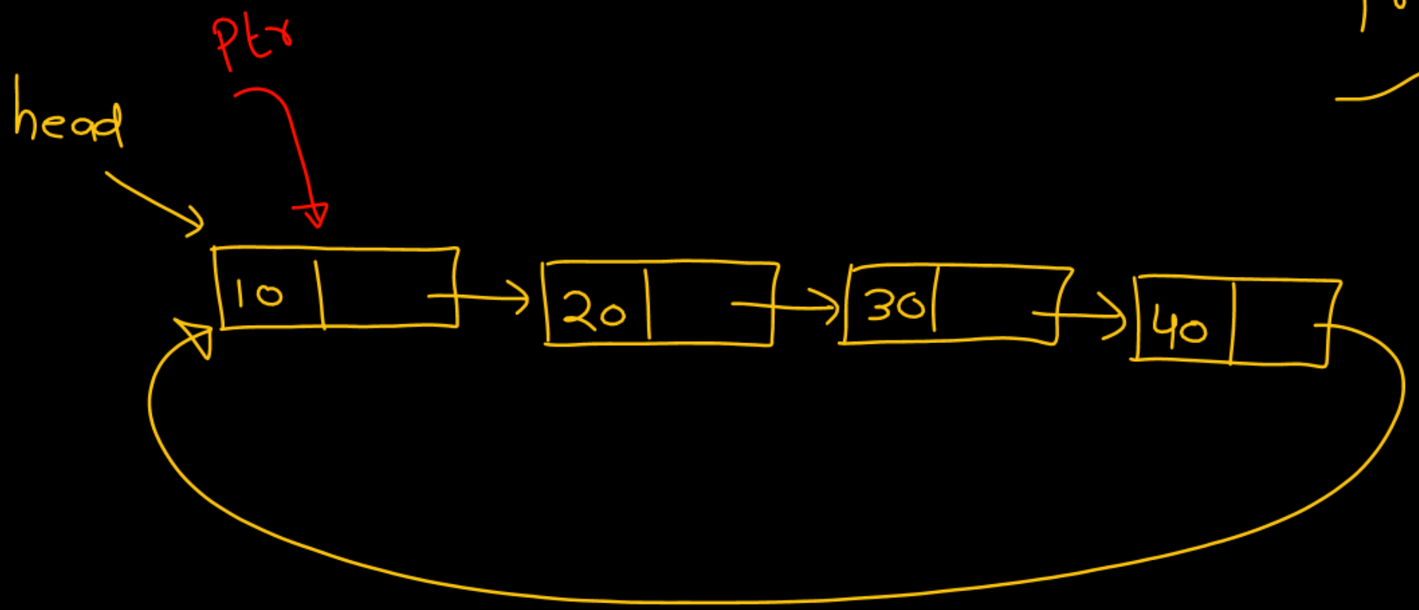


(i) Ptr → Prev → next = Ptr → next

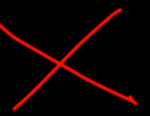
(ii) Ptr → next → Prev = Ptr → Prev

free (Ptr)

Circular Linked List

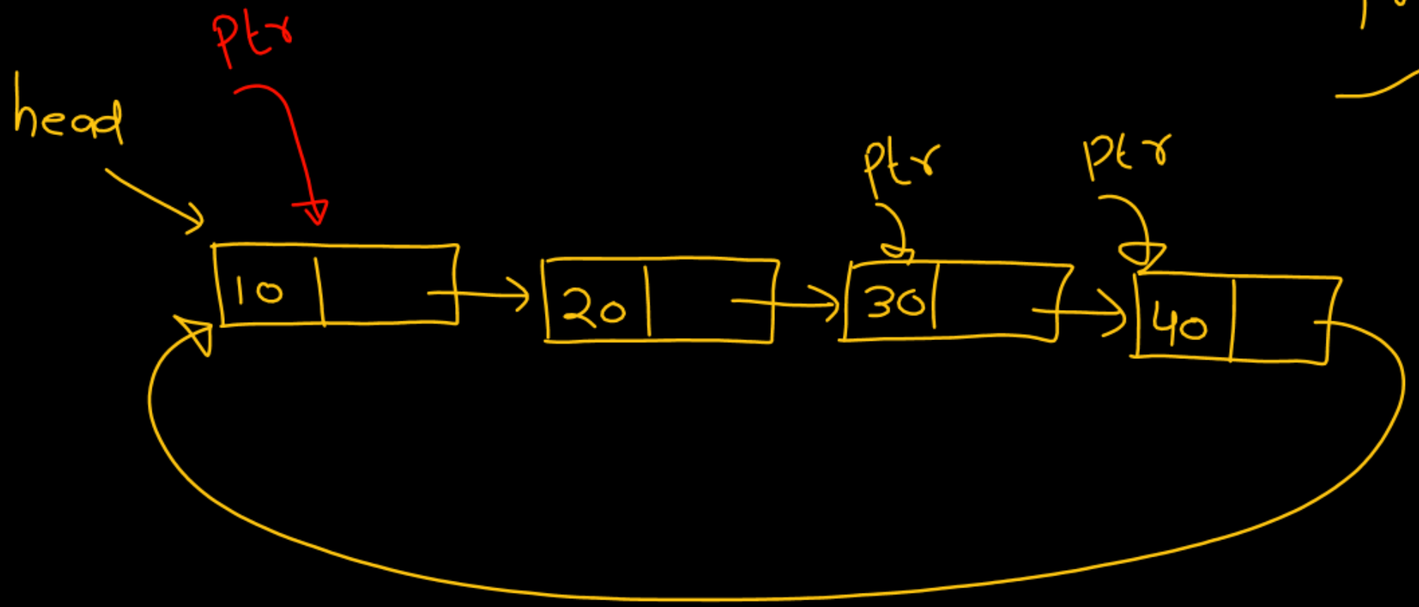


Traversal code



```
while (ptr != head)
{

}
```

Circular Linked List

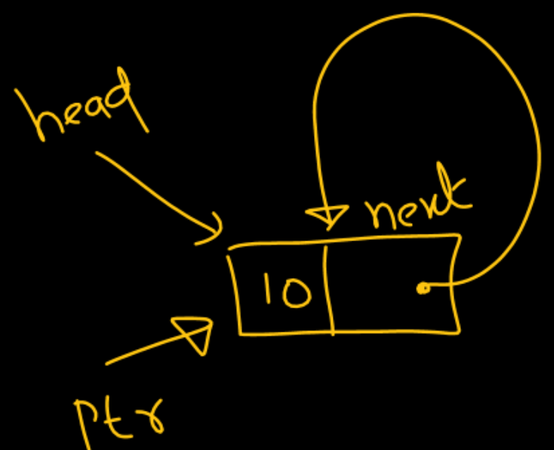


10 20 36 40

Traversal code

```
if (head == NULL)
    return;

do {

pf("%d", Ptr->data);
Ptr = Ptr->next;

} while(Ptr != head)
```

Circular Linked List

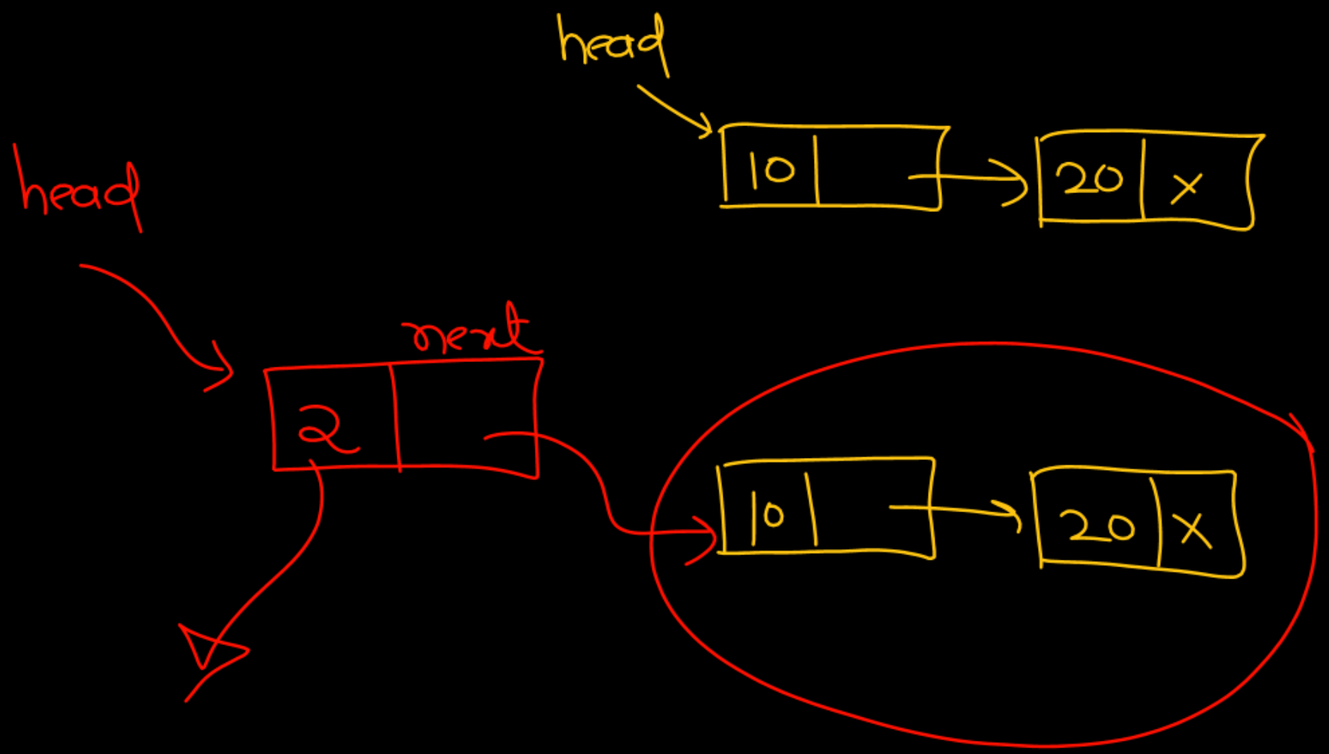


10

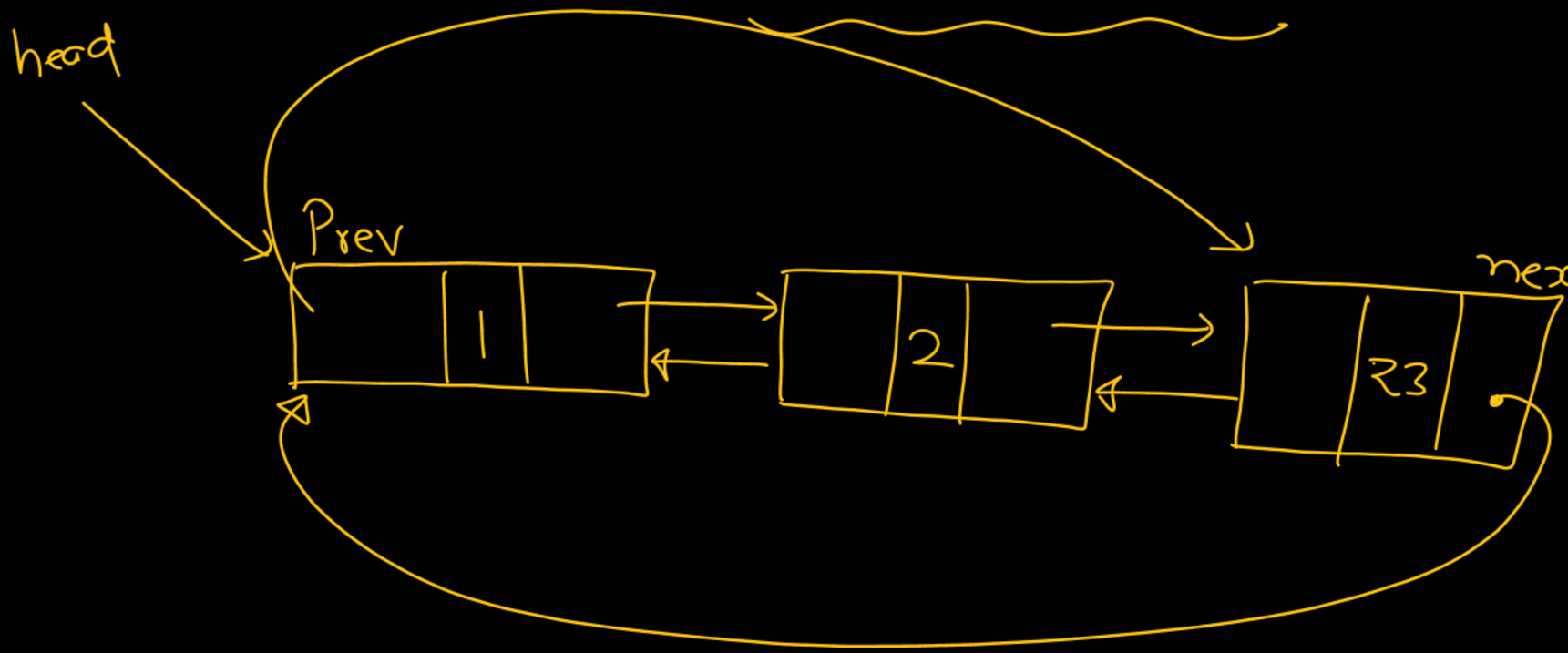
Traversal code

```
if (head == NULL)
    return;

do {

bf ("%d", Ptr->data);
Ptr = Ptr-> next ;


} while(Ptr != head)
```

Header Linked List



Doubly Circular LL



Traversal
deletion
insertion
Search
Merge
Reverse

13hrs


THANK YOU GW
SOLDIERS !