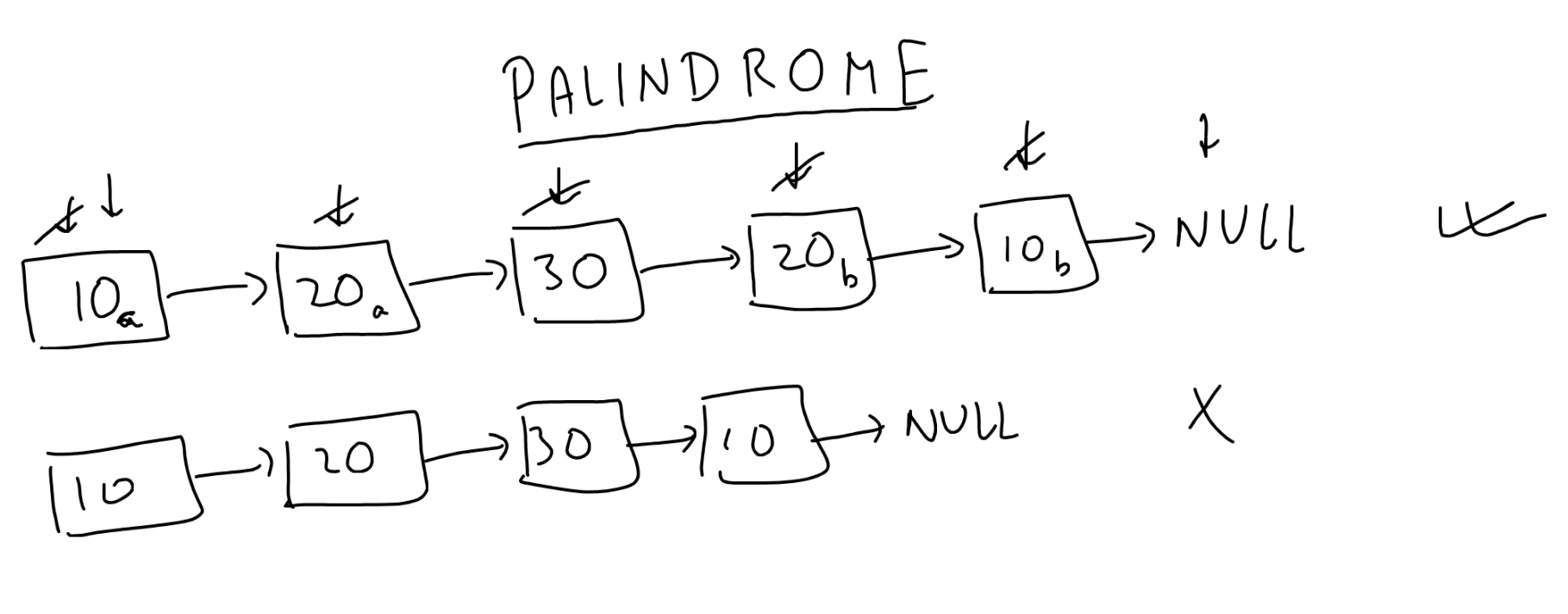
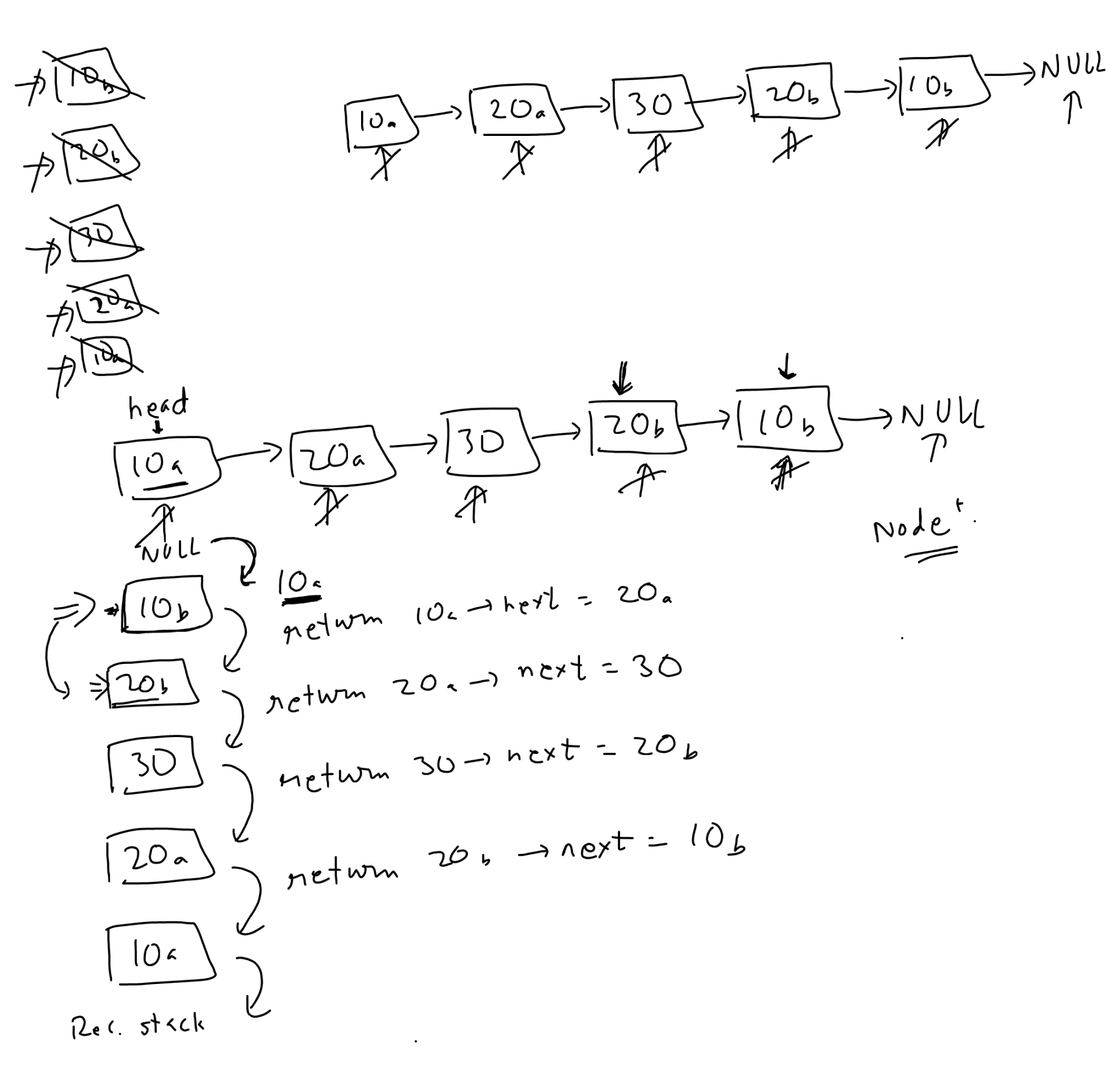
PALINDROME

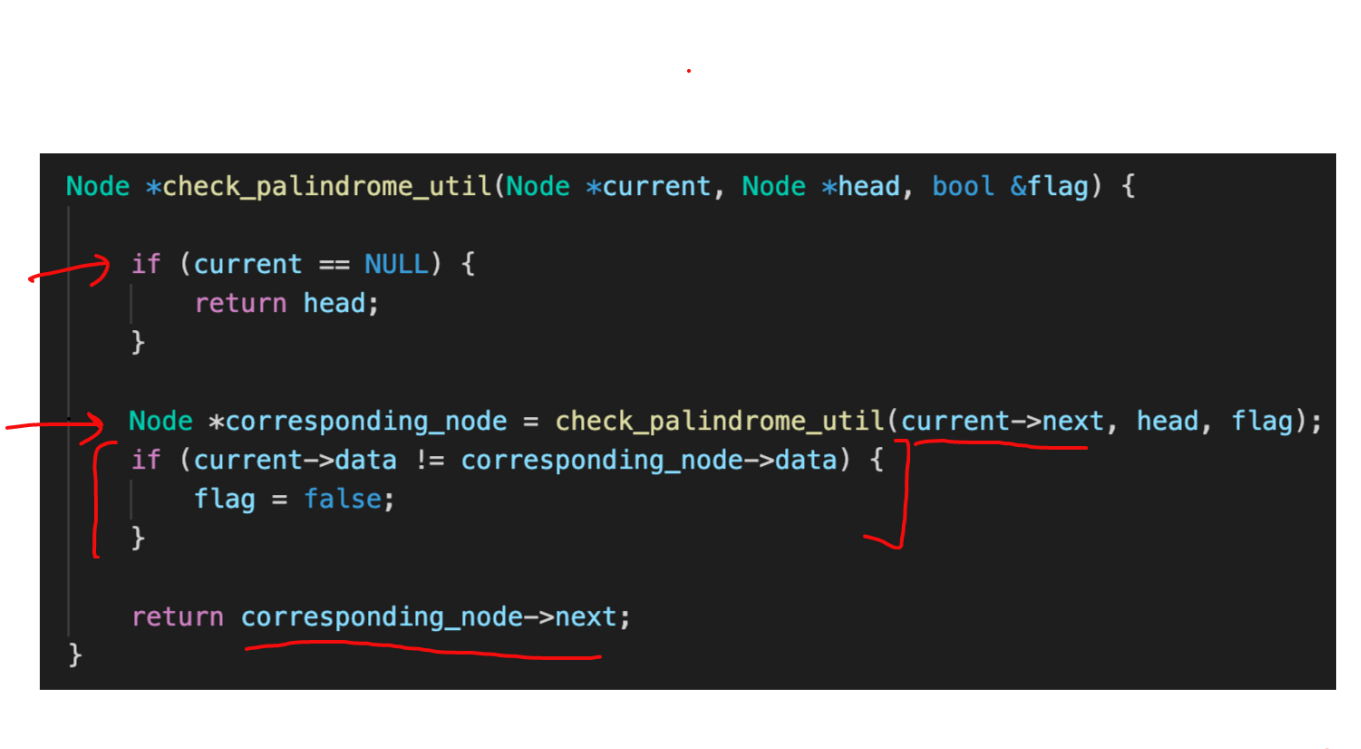

10a
20a
30
20b
10b
NULL
10
20
30
10
NULL
X
10
20
30
20
10

Create duplicate reverse:

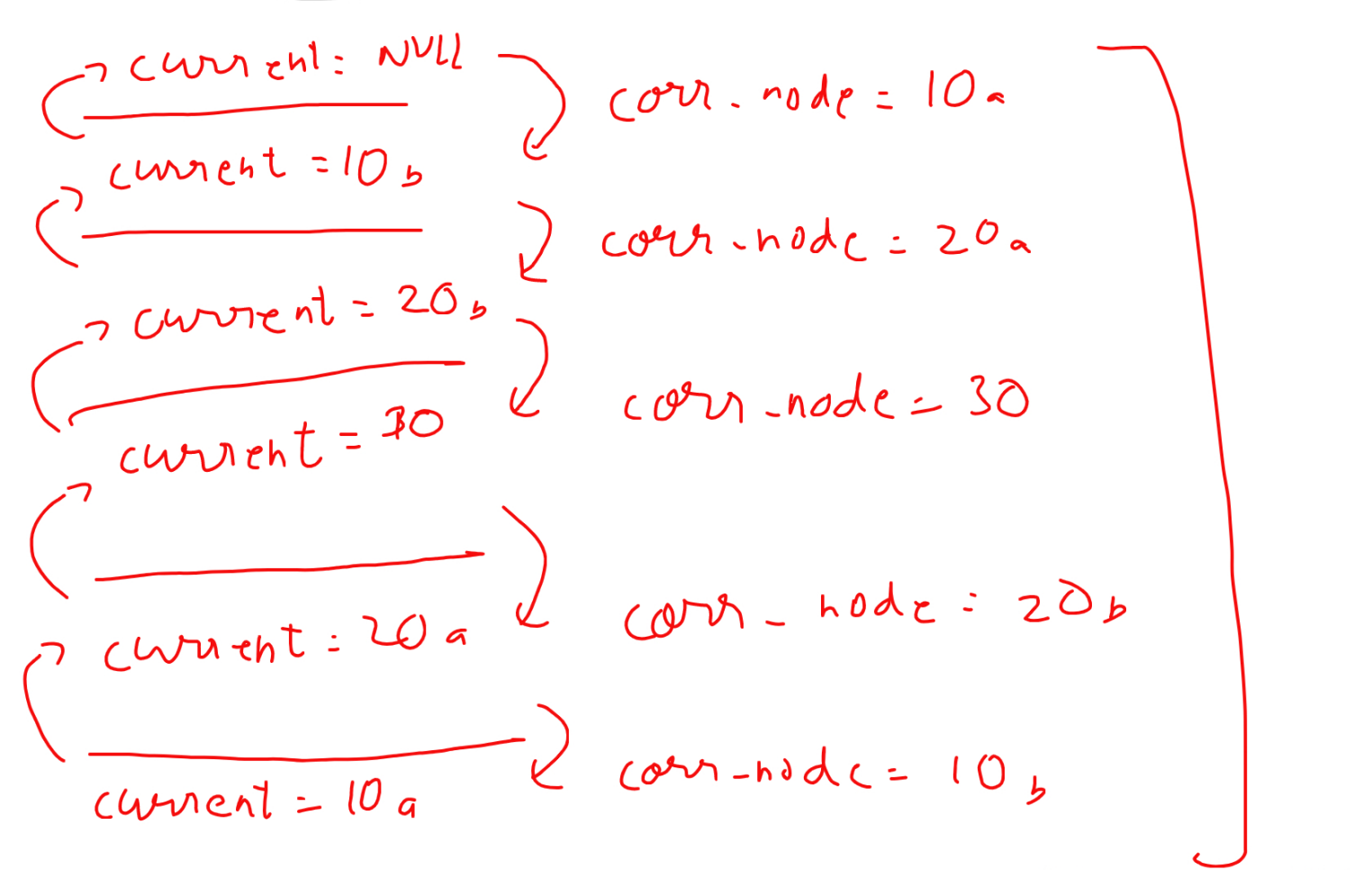
TC: O(n) (2 traversals)

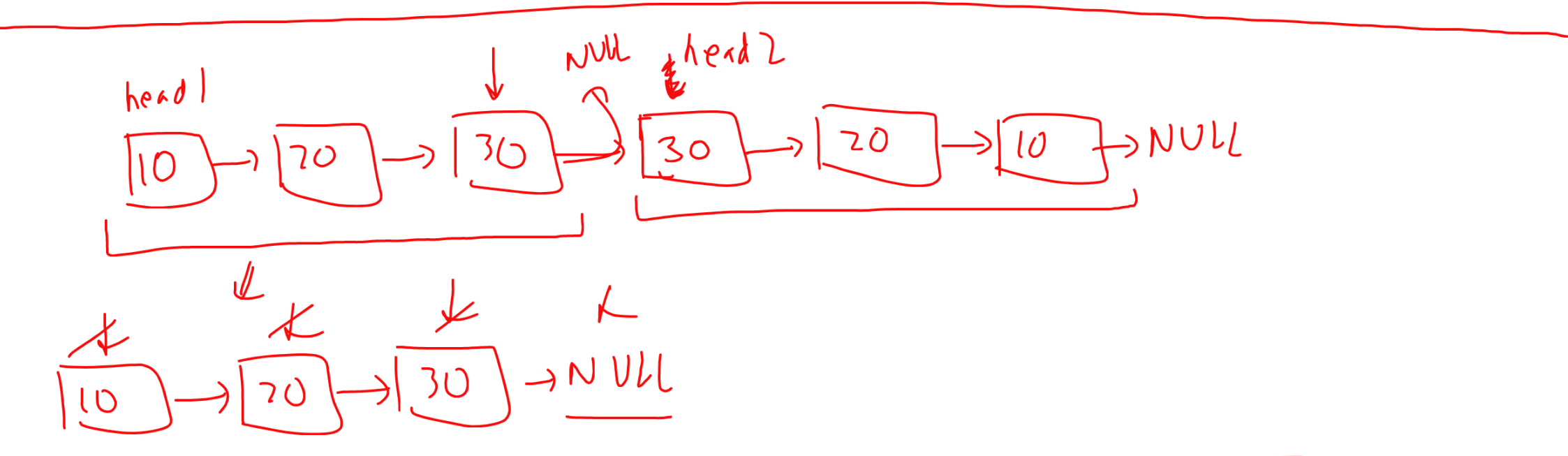
Aux.space: O(n)


10a
20a
30
20b
10b
NULL
head
Node*
NULL
10a
return 10a → next = 20a
return 20a → next = 30
return 30 → next = 20b
return 20b → next = 10b
Rec. stack



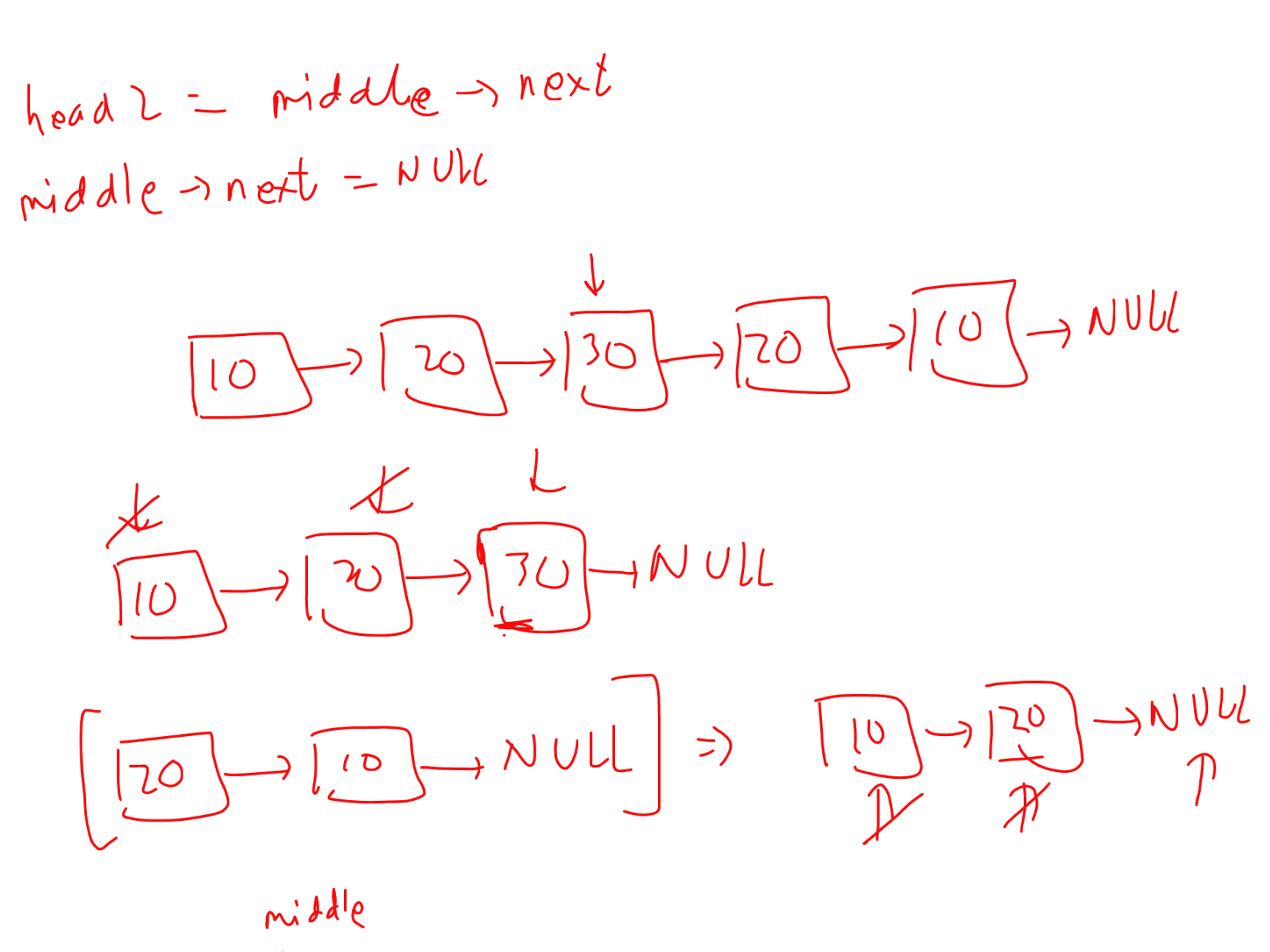
```
Node *check_palindrome_util(Node *current, Node *head, bool &flag) {
    if (current == NULL) {
        return head;
    }

    Node *corresponding_node = check_palindrome_util(current->next, head, flag);
    if (current->data != corresponding_node->data) {
        flag = false;
    }

    return corresponding_node->next;
}
```

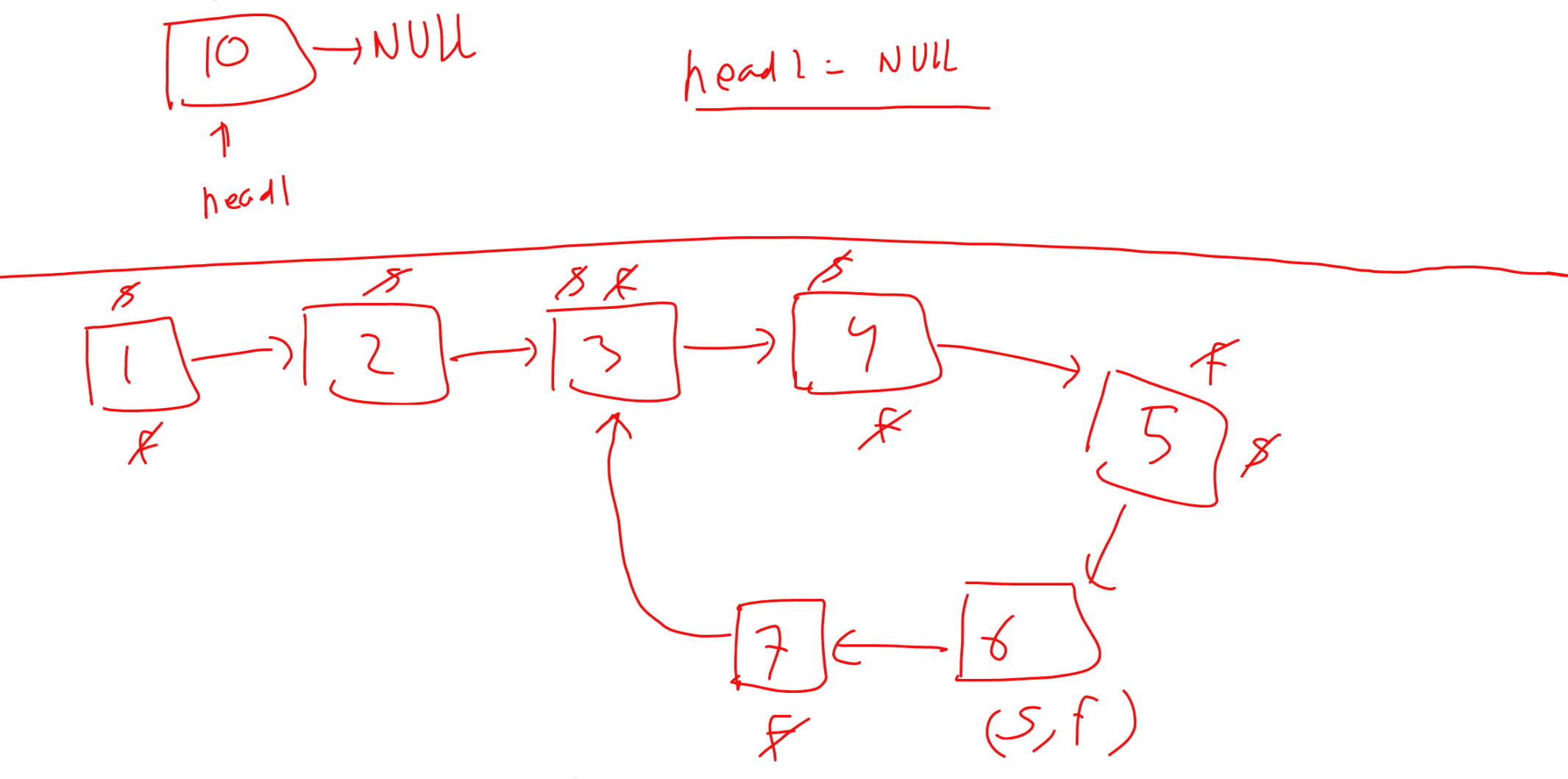

10a
20a
30
20b
10b
NULL
current: NULL
current = 10b
current = 20b
current = 30
current : 20a
current = 10a
corr. node = 10a
corr. node = 20a
corr. node = 30
corr. node : 20b
corr. node = 10b


head1
head2
NULL
10
20
30
20
10
NULL
rev
30
20
10
NULL

head2 = middle → next

middle → next = NULL

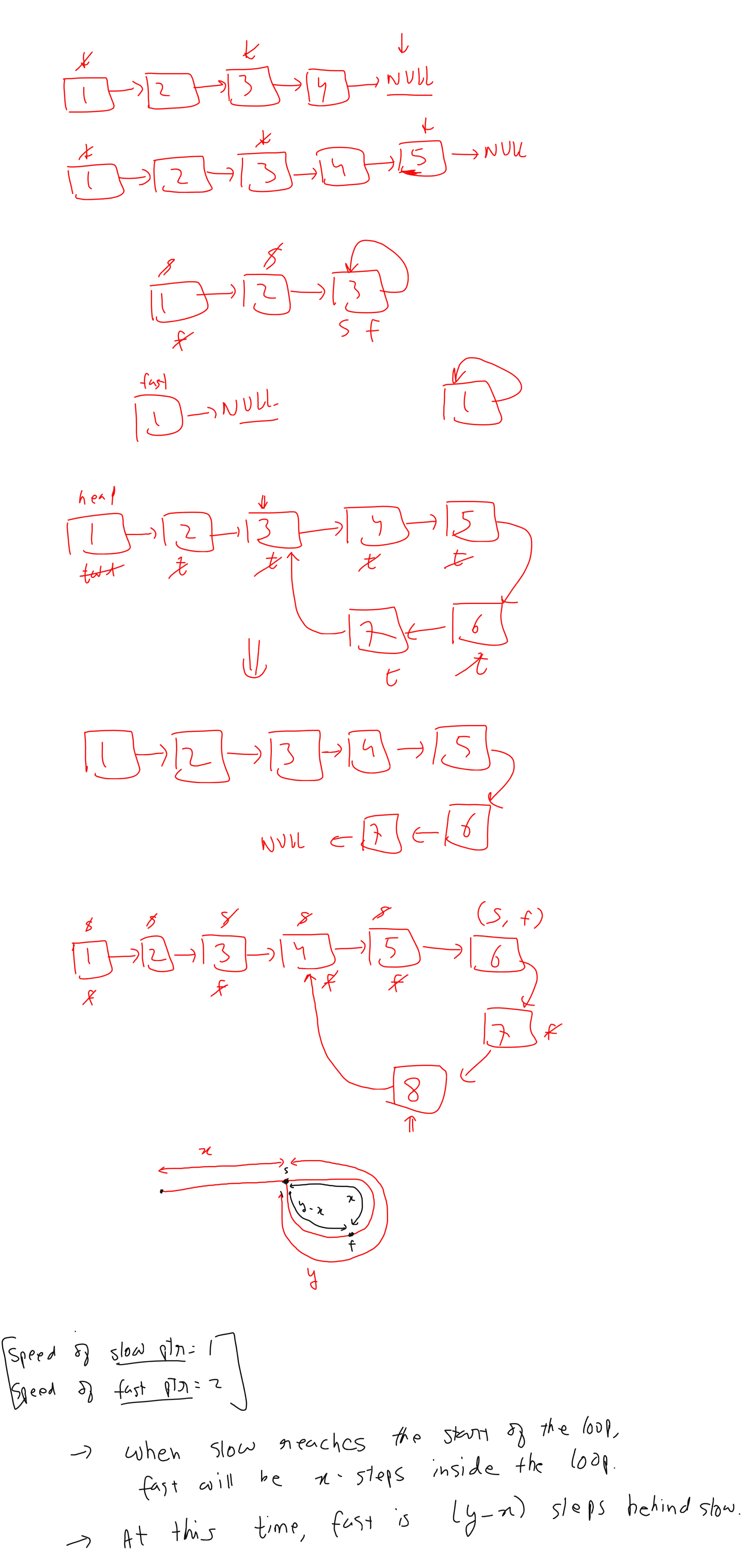

10
20
30
20
10
NULL
20
10
NULL
middle
10
NULL
head1
head1 = NULL



#1 Hashmap-Based approach:

TC: O(n)

Aux.space: O(n)


1
2
3
4
5
NULL
s f
fast
1
NULL
head
6
7
8
(s, f)
x
y
y-x

Speed of slow ptr = 1
Speed of fast ptr = 2

→ when slow reaches the start of the loop, fast will be x-steps inside the loop.

→ At this time, fast is (y-x) steps behind slow.

→

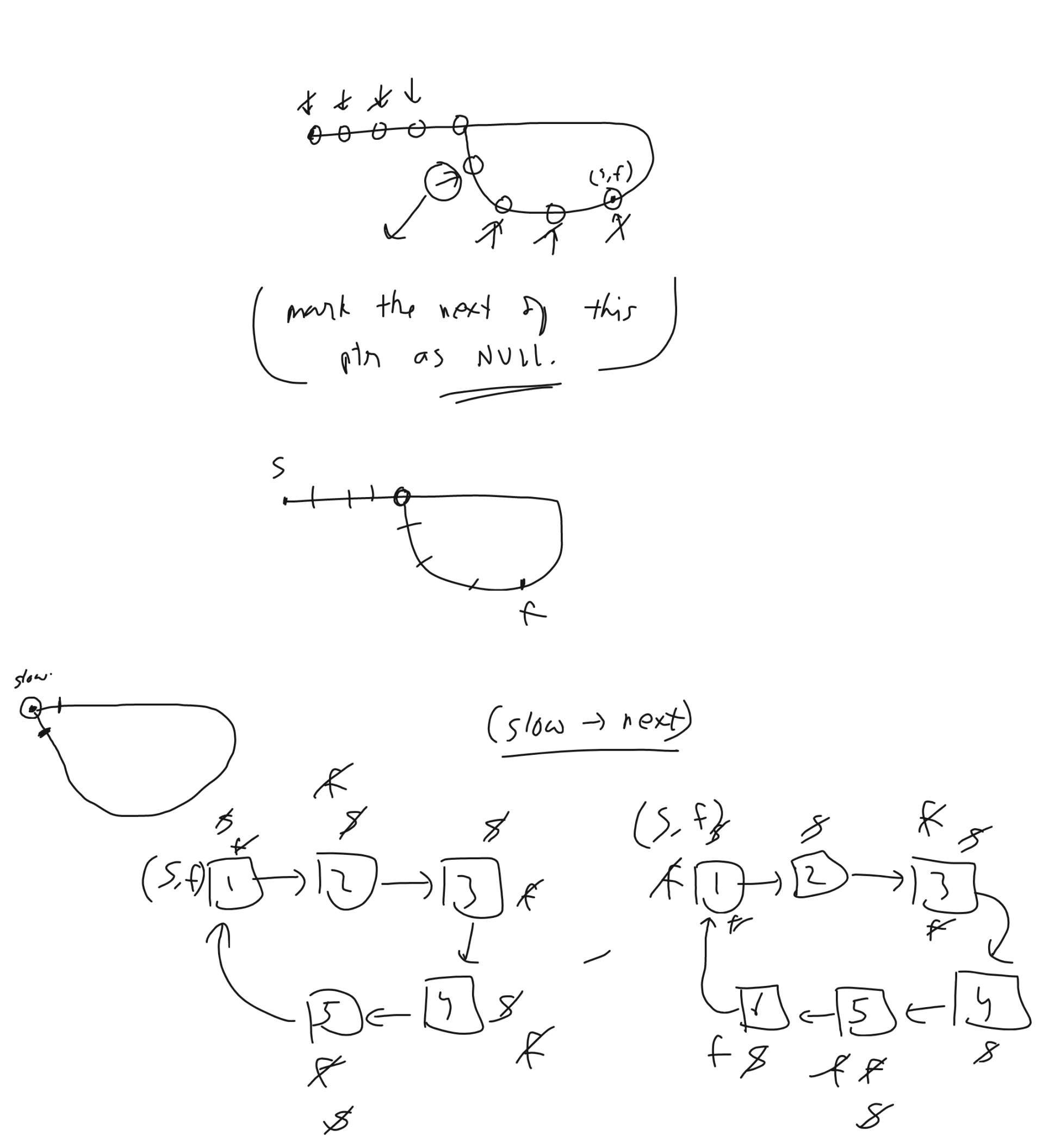
Speed = Distance / Time ⇒ Time = Distance / Speed

Relative distance b/w fast & slow = (y-x)

Relative speed = (2-1)

∴ Time taken by fast ptr to catch the slow ptr:

⇒ (y-x) / (2-1) = (y-x)


x
y-x
(s, f)
x
(s, f)
mark the next of this ptr as NULL
s
f
slow
(slow → next)
(s, f)
1
2
3
4
5
6

---

int → Base-10 numbers → Decimal numbers (0, 1, 2, 3, ..., 9)

Base-2 numbers → Binary numbers (0, 1)

FACT: Every decimal no. has a distinct Binary representation.

32-bit integers   64-bit integers

0 1 2 3 ... 31   0 1 2 3 ... 63

2^{32} 2^{64}

	2^3	2^2	2^1	2^0	
0 →	0	0	0	0	
1 →	0	0	0	1	
2 →	0	0	1	0	
3 →	0	0	1	1	$2^0 + 2^1$
4 →	0	1	0	0	
5 →	0	1	0	1	$2^2 + 2^0$
6 →	0	1	1	0	
7 →	0	1	1	1	$2^2 + 2^1 + 2^0$
8 →	1	0	0	0	
9 →	1	0	0	1	
10 →	1	0	1	0	

1010

$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$

= 8 + 0 + 2 + 0

= 10

	Quot.	Rem
10/2	5	0
5/2	2	1
2/2	1	0
1/2	$\boxed{0}$ STOP	1

1010

→ BITWISE OPERATORS

1) BITWISE AND (&) –

I/P		O/P	
0	0	0	← unset-bit
0	1	0	
1	0	0	
1	1	1	← set-bit

2) BITWISE OR (|)

I/P		O/P
0	0	0
0	1	1
1	0	1
1	1	1

3) BITWISE XOR (^)

I/P		O/P
0	0	0
0	1	1
1	0	1
1	1	0

4) RIGHT-SHIFT (>>)

2^3 2^1 2^0
1010 → (10)
(>>) 2^2 2^0
0101 → (5)

(floor division by 2)

5) LEFT-SHIFT (~~>>~~) (<<)

2^2 2^0
0101 → (5)
(<<) 2^3 2^1
1010 → (10)

(Multiplication by 2)

a ^ a = 0
a ^ 0 = a

a = 9
  1001
^ 0000
  1001

COUNT SET BITS

n = 10 ⇒ 2
n = 7 ⇒ 3

n = 10 (1010)
n & 1 = 0
n = (n >> 1)

n = (0101)
n & 1 = 1 ⇒ cnt++
n = (n >> 1)

n = (0010)
n & 1 = 0
n = (n >> 1)

n = (0001)
n & 1 = 1 ⇒ cnt++
n = (n >> 1)

n = (0000)
STOP

(n & 1) ⟶ = 1 if the rightmost bit is set

  1010
& 0001
  0000 = 0 otherwise.

O(no. of bits)

---

n = 10
n-1 = 9

  1010
& 1001
  1000 = (8)

n = 8
n-1 = 7

  1000
& 0111
  0000 = 0 (STOP)

No. of operation till the number became zero =
No. of set bits

$[n = n \,\&\, (n-1)]$

10000000

n = 11
n-1 = 10

  1011
& 1010
  1010 = (10)

n = 10
n-1 = 9

1010
1001
1000 = (8)

n = 8
n-1 = 7

1000
0111
0000 = (0) (STOP)

no. of set bits = 3

num = 1010111100100
num - 1 = 1010111100011
num & (num-1) = 1010111100000

K = 2

n = 10 , K =
⇓
1010
3rd 0th

K	O/P
K = 0	False
K = 1	True
K = 2	False
K = 3	True

n = 1010
mask = & 0010
0010 ⇒ 1st bit set

n = 1010
mask = & 0100
0000 ⇒ 2nd bit unset

1 = 0001
(1<<1) = 0010
(1<<2) = 0100
(1<<3) = 1000