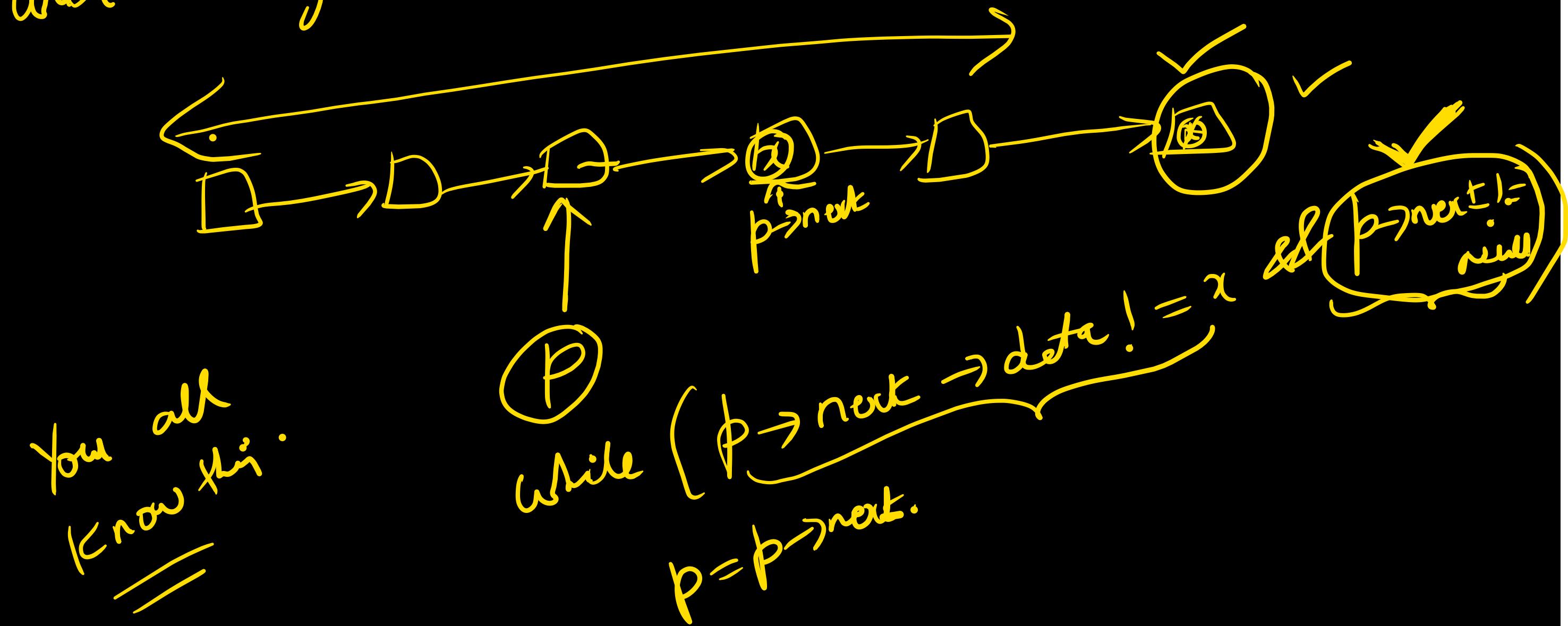
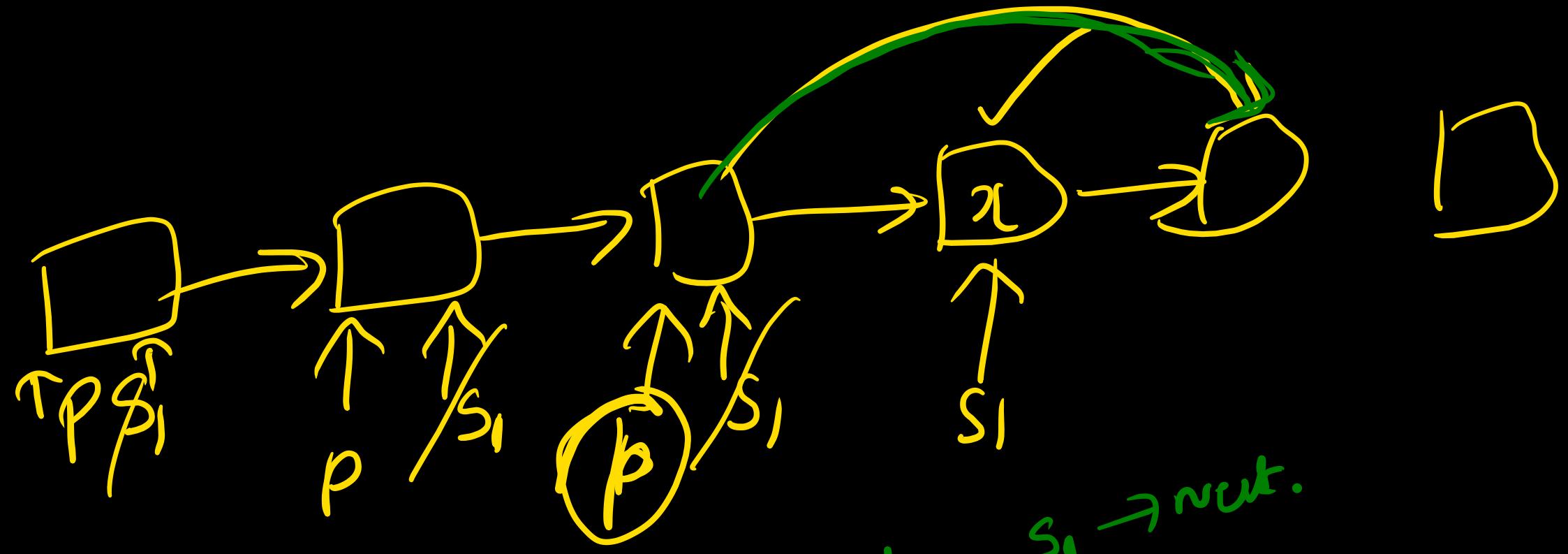


Write an algo to delete node with data 'x' in SLL.

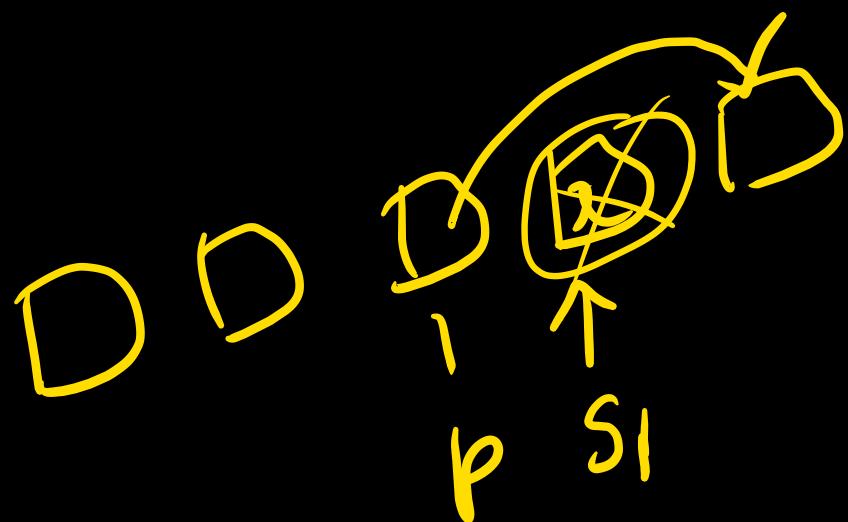


Two pointers



$p \rightarrow \text{Nuk} = S_1 \rightarrow \text{nuk.}$
 $\text{pre}(S_1)$

Outer Cons: if ($S == \text{null}$) (pf "no x " in SLL)



$S_1 = S$ F
while ($S_1 \rightarrow \text{data} != x$) && $S_1 \rightarrow \text{next} != \text{null}$)
{ $p = S_1;$ } $\rightarrow p$ in following S_1
 $S_1 = S_1 \rightarrow \text{next}$

}
if ($S_1 \rightarrow \text{data} == x$)
{ $p \rightarrow \text{next} = S_1 \rightarrow \text{next};$ free(S_1)
return (S);
}

End

end in rec'd

else

{ if ($s_1 \rightarrow \text{data} \leq x$)

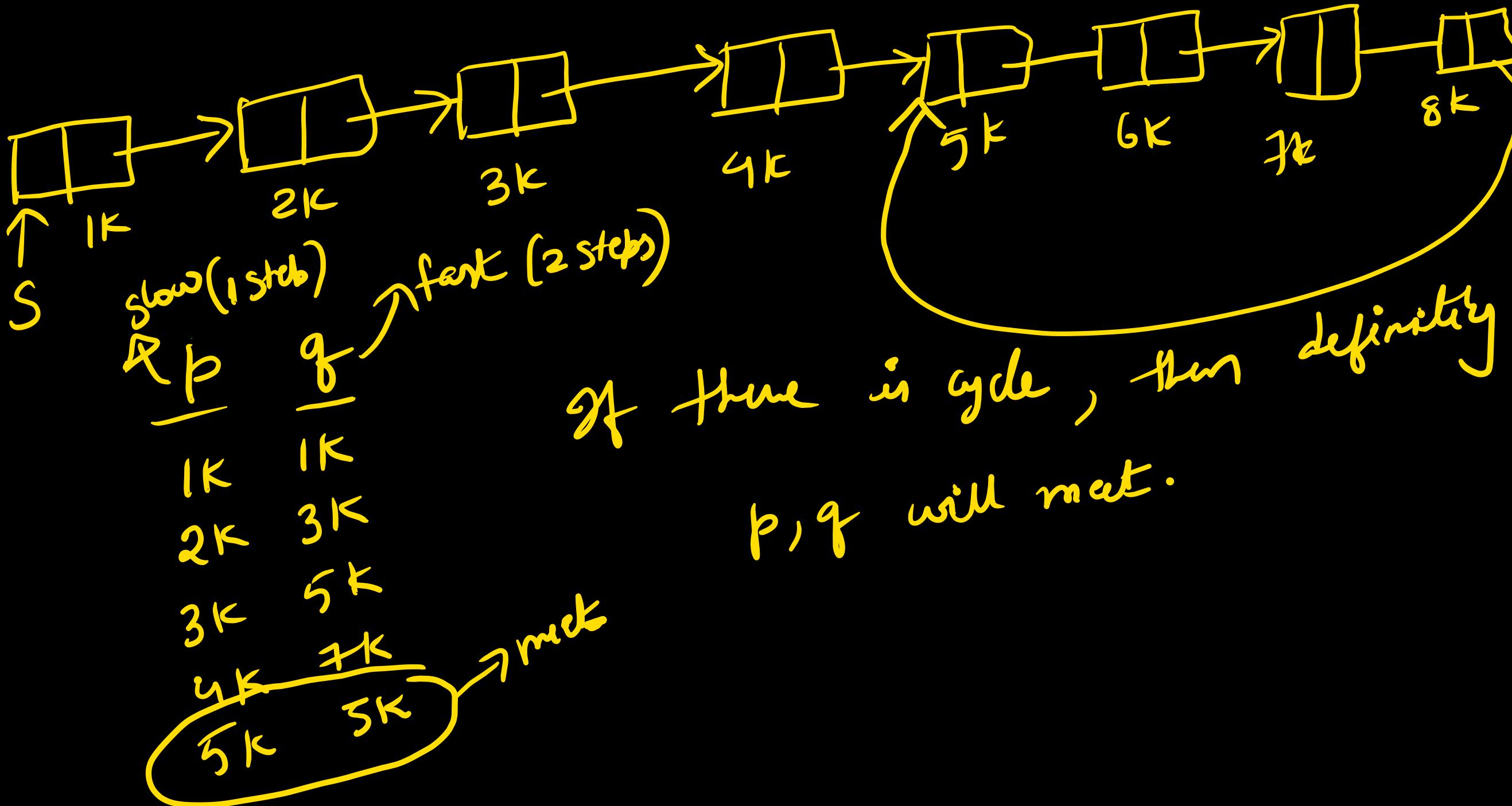
$p \rightarrow \text{next} = 10$

$\text{free}(s_1)$
 $\text{return}(s)$





Write an algo to detect cycle or loop in a given SLL.

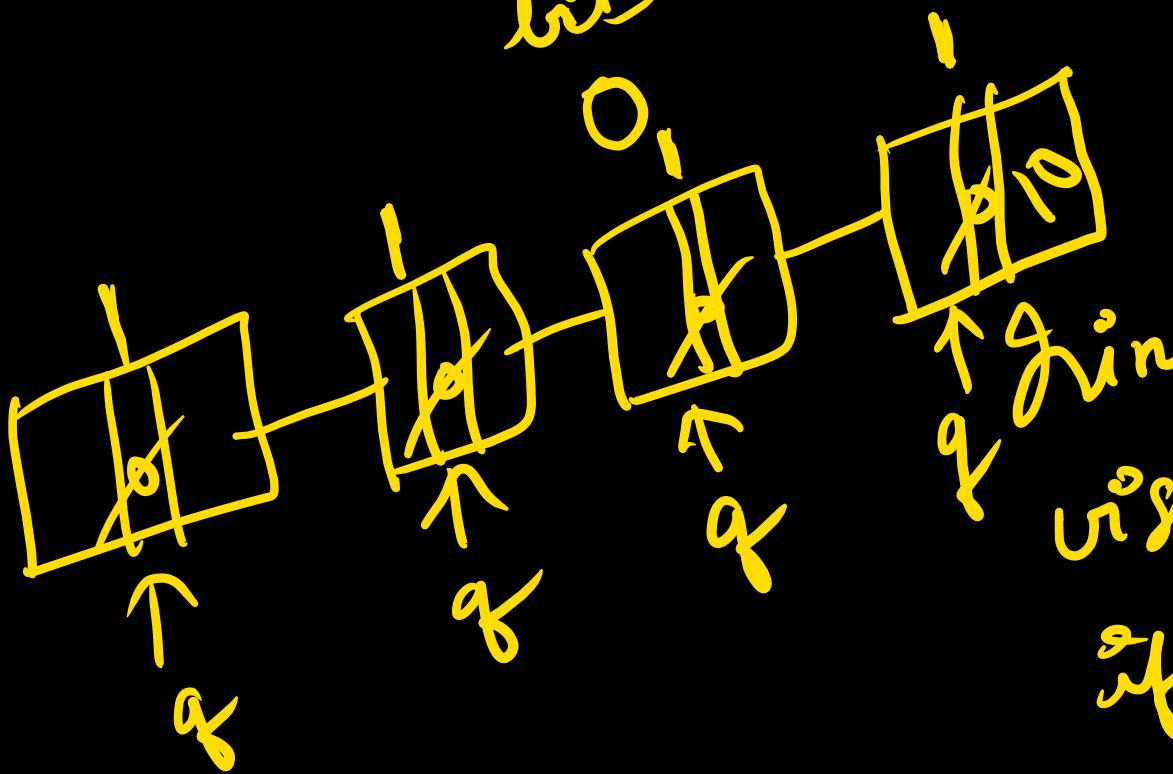
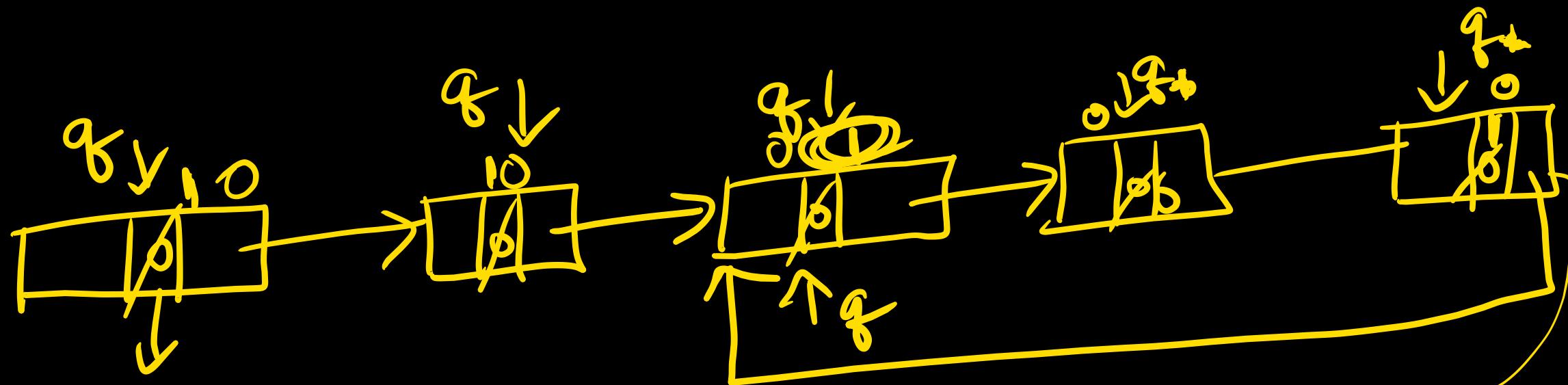


Gate i
 $p = q = s$
 $\text{while } (q \neq \text{null} \text{ & } q \rightarrow \text{next} \neq \text{null} \text{ & } q \rightarrow \text{next} \rightarrow \text{next} \neq \text{null})$

```

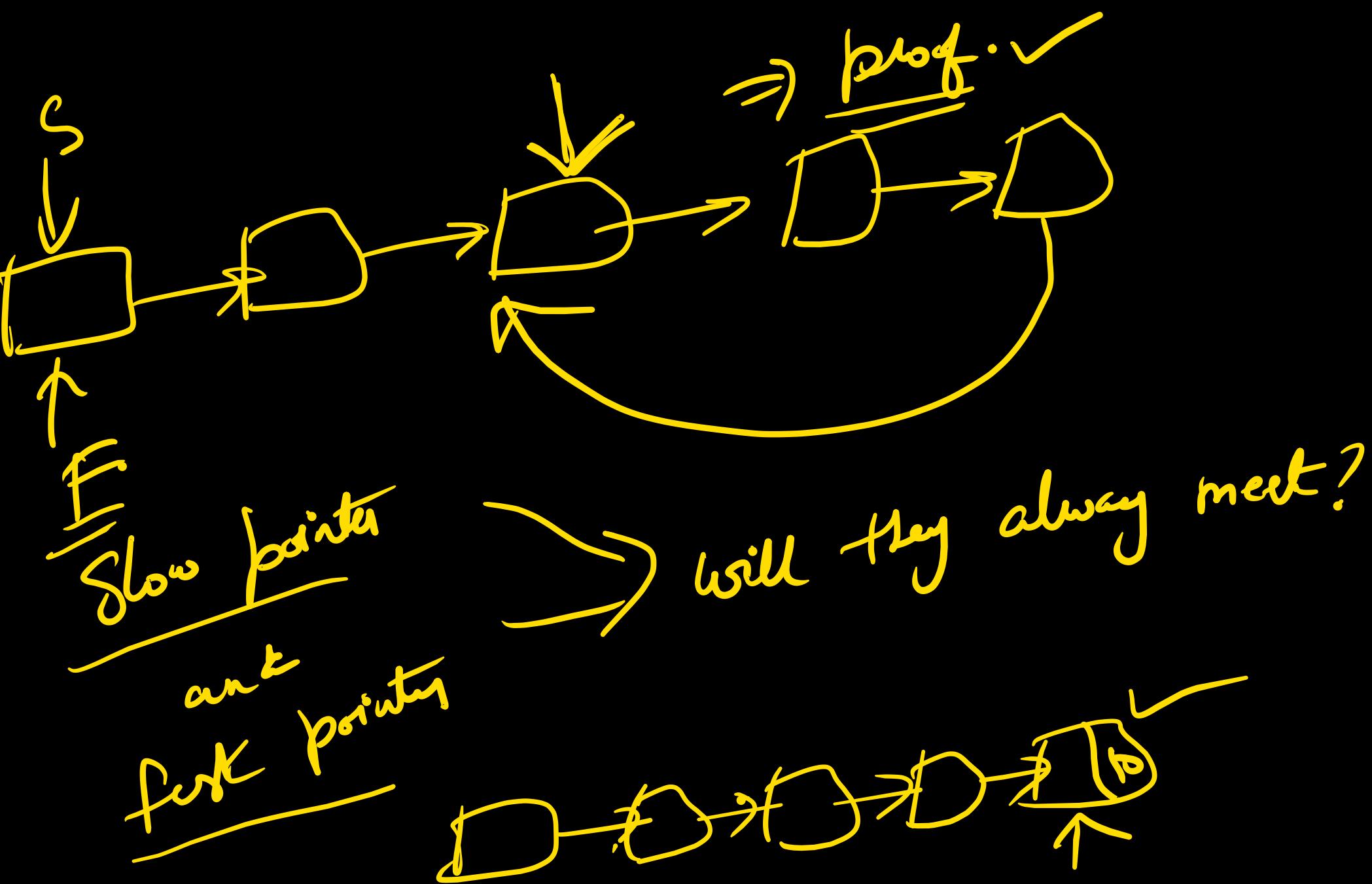
    {
        p = p → next;
        q = q → next → next;
    }
    if (p == q)
        pf (loop found);
    return algo;
}
pf (no loop found)
    
```

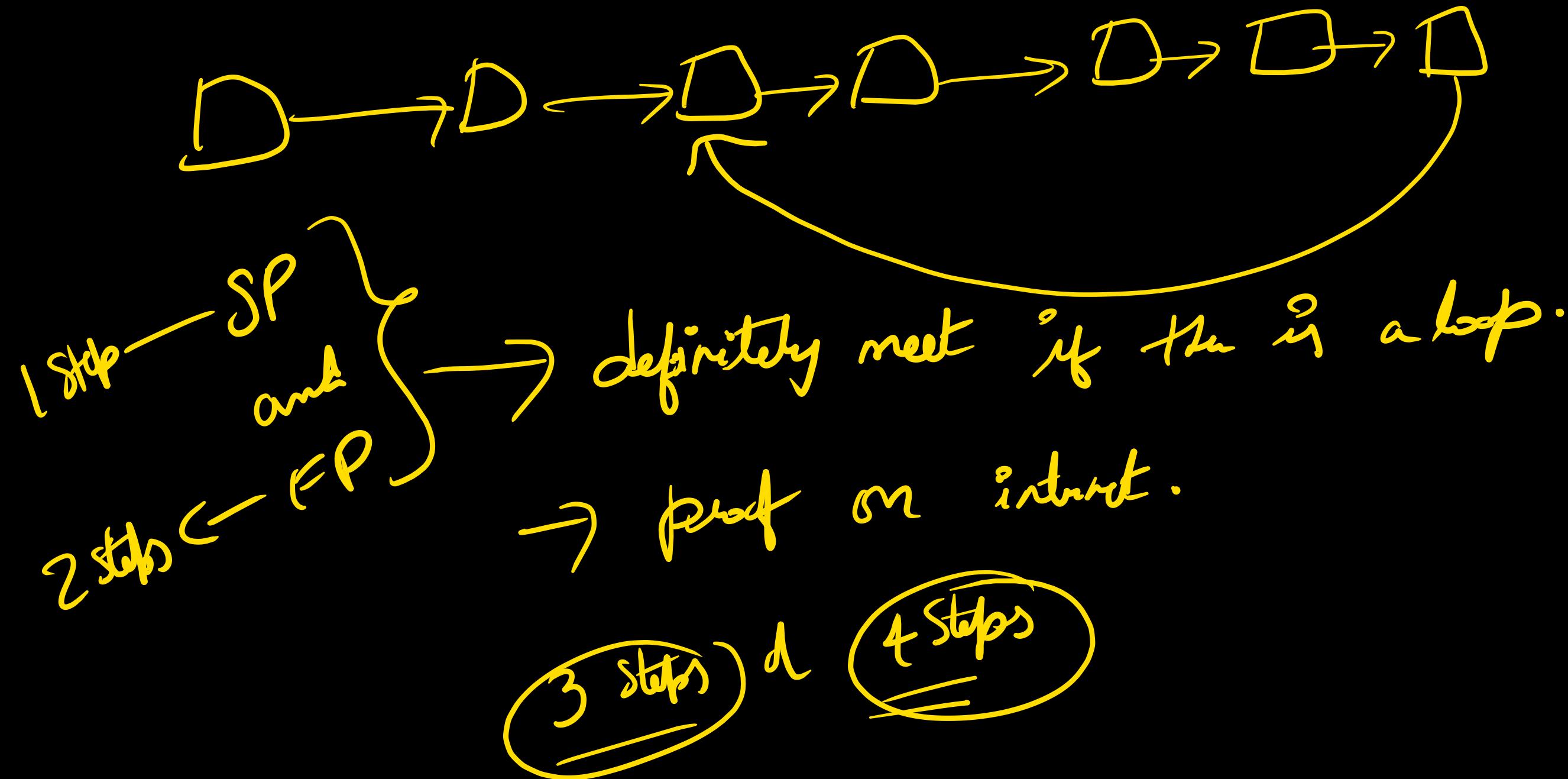
In gate example
 what is line ①
 add a new condition
 if $s == \text{null}$
 if $s \rightarrow \text{next} == \text{null}$.



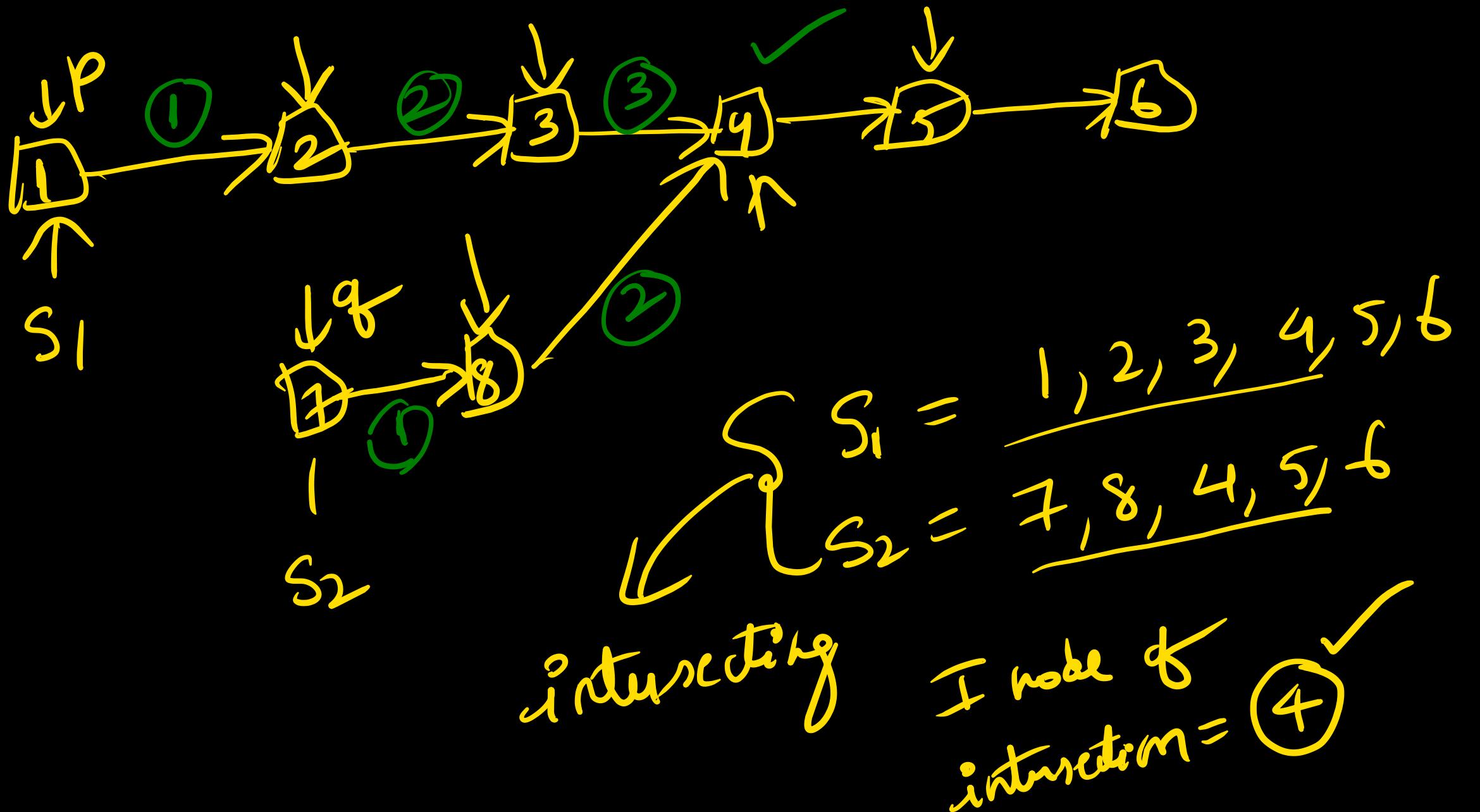
init all 0's.
 visit all the nodes and make 0's 1's
 if you come across 1, loop in there.

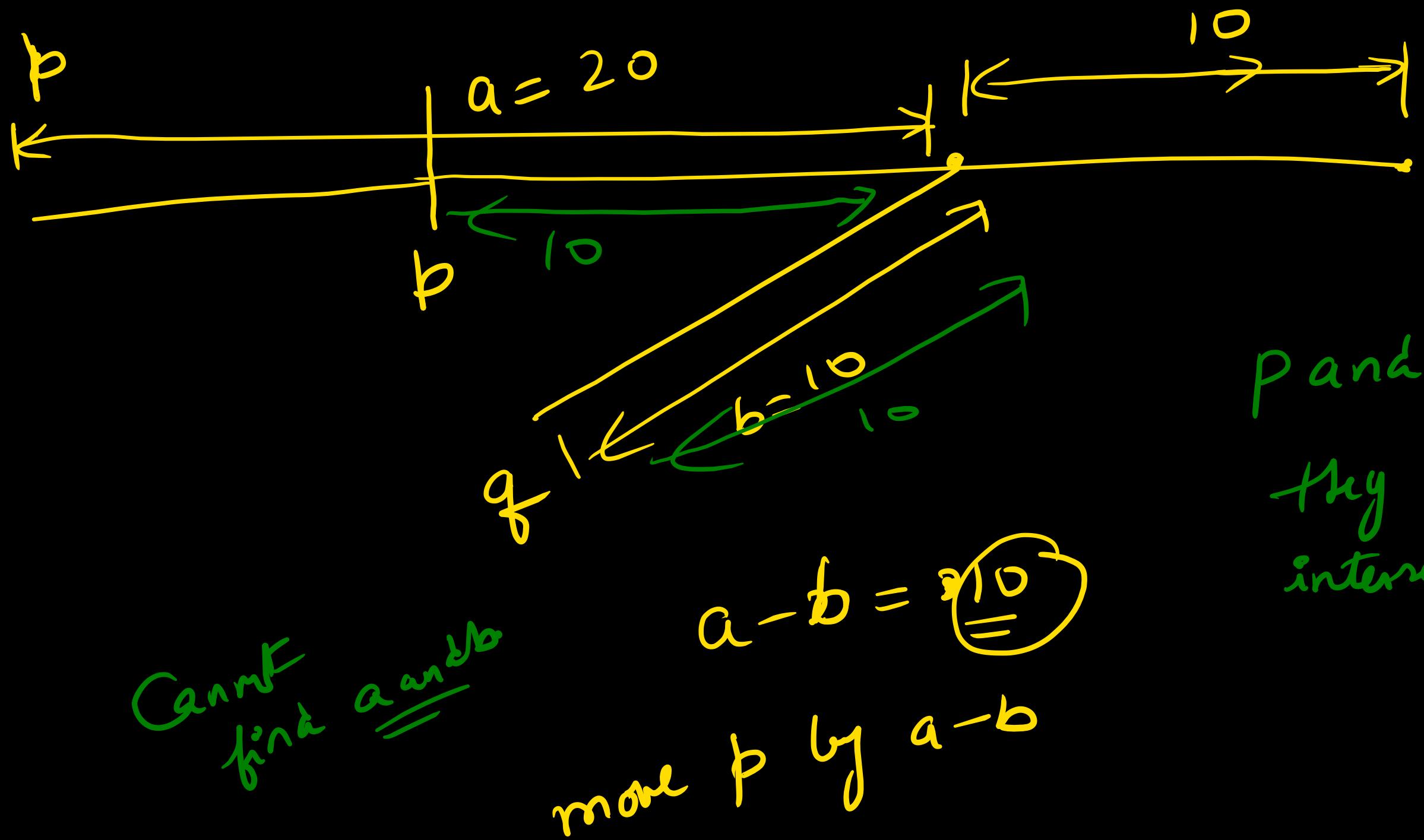
$SC = O(n)$; extra
 space is
 required.

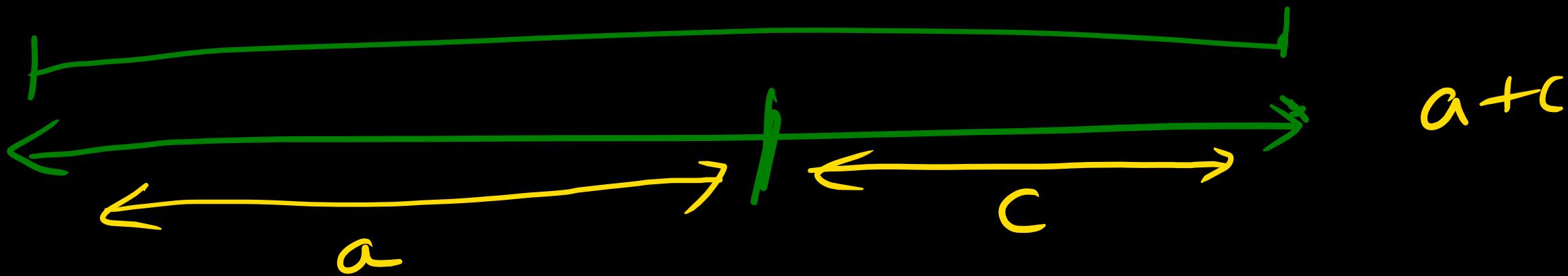


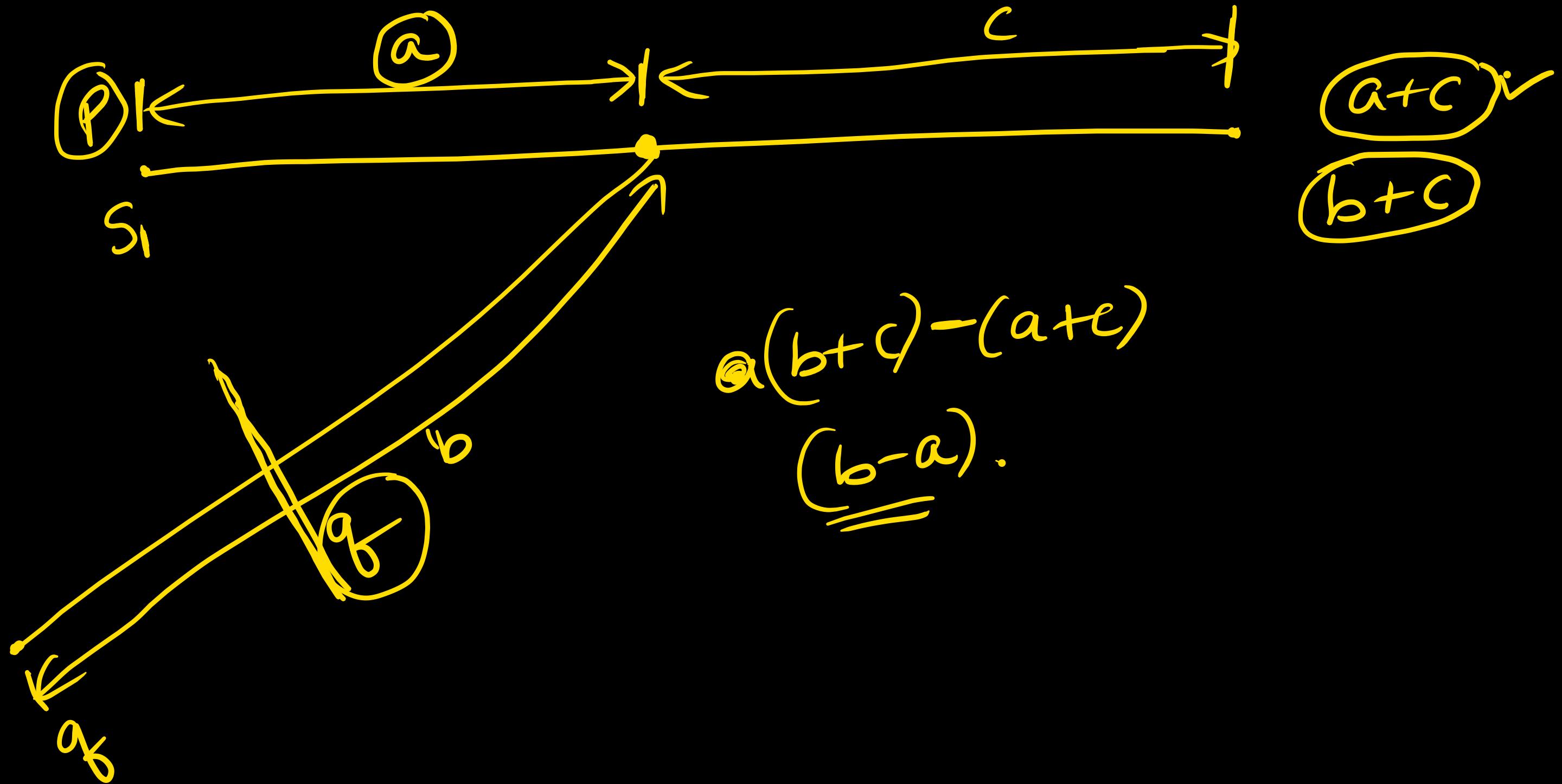


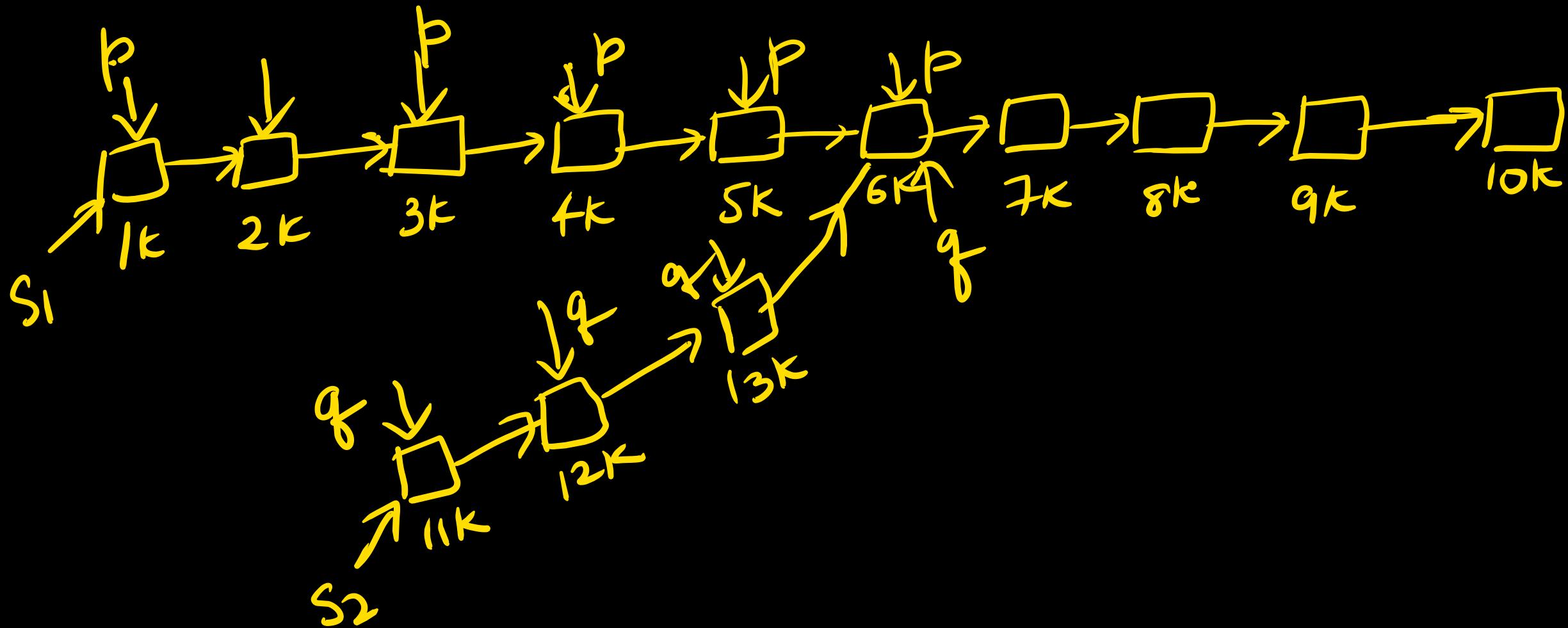
Find the address of I intersecting node.











$$|S_1| = 10 \quad 10^{-8} = 2$$

$$|S_2| = 8$$

~~Do~~ $m = \text{size of } \underline{\underline{LL1}}$, $n = \text{size of } \underline{\underline{LL2}}$

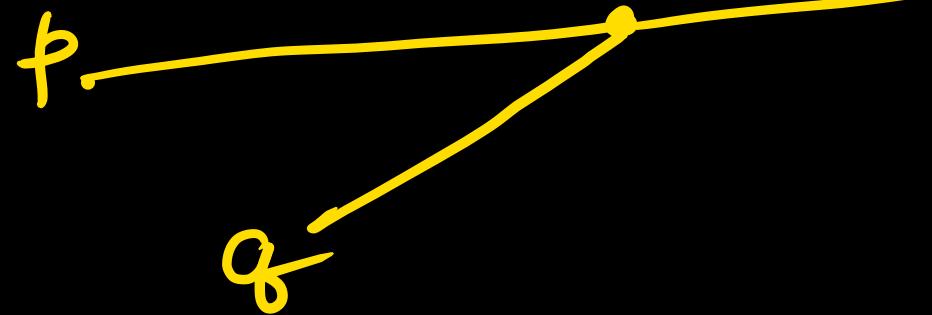
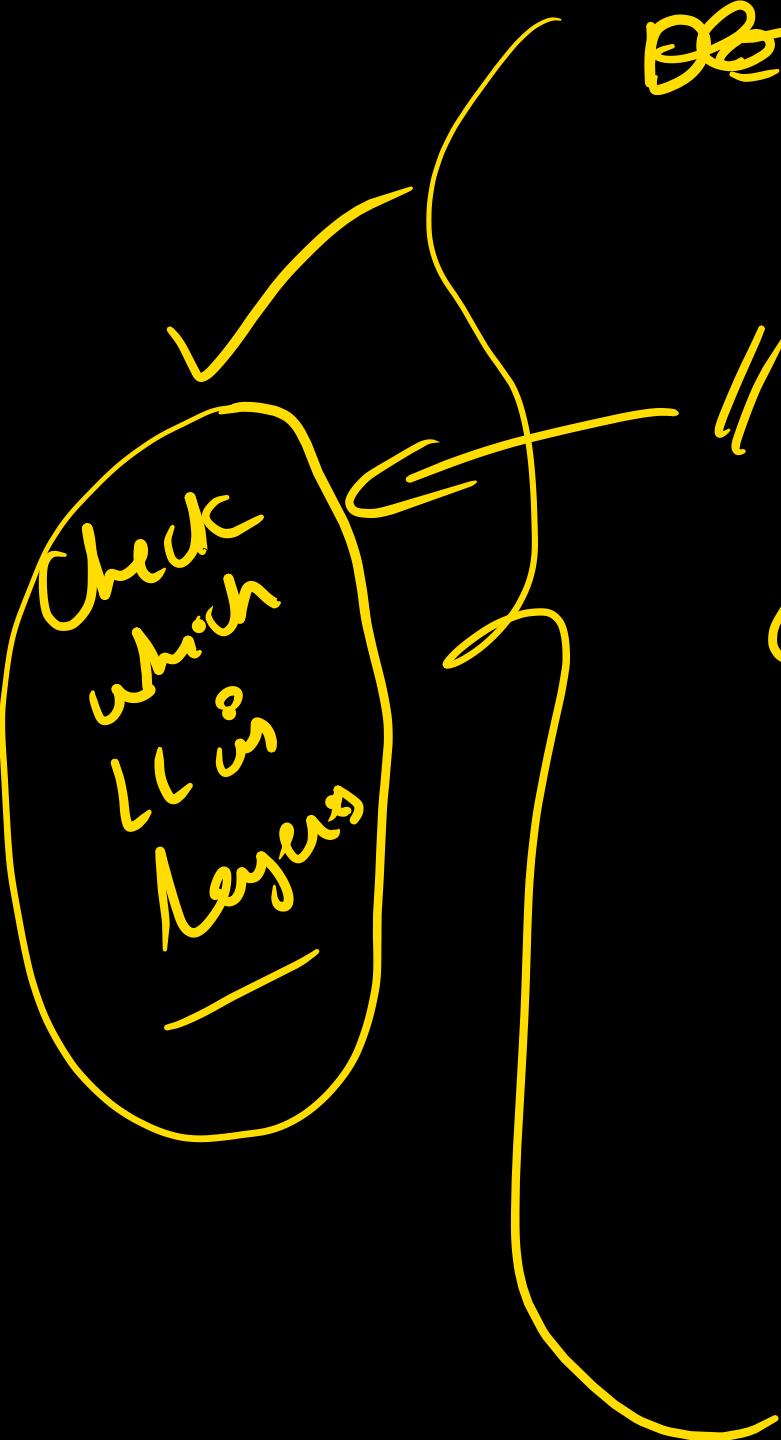
$$D = |m - n|$$

// move the pointers on the layer LL

while ($D \neq 0$)

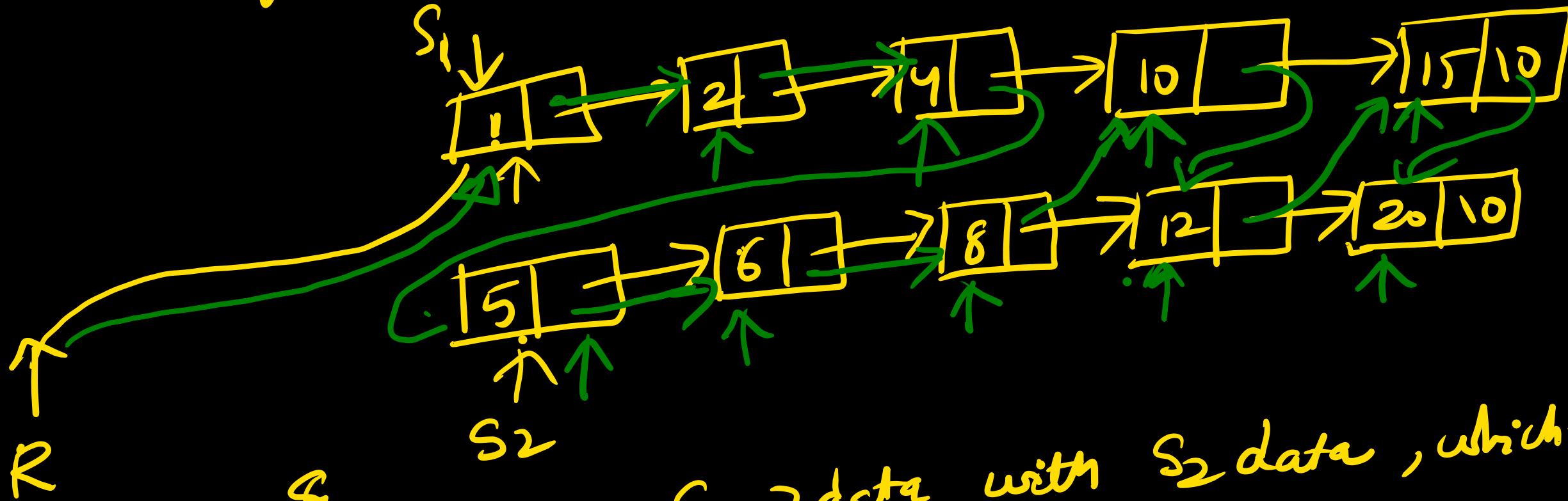
{
 ~~p = p->next~~
 $D = D - 1$

}
while ($p \neq q$)
{
 $p = p->next$; $q = q->next$;
}

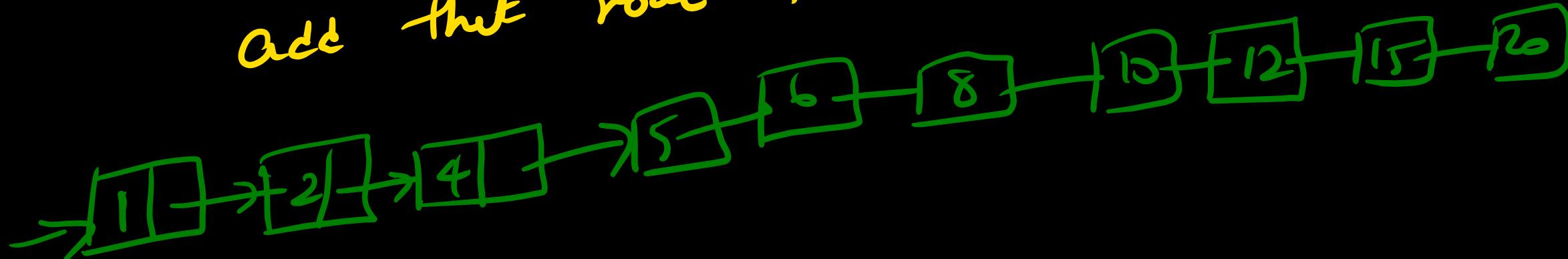


(corner cases.)

merge two sorted LL into one linkedlist.

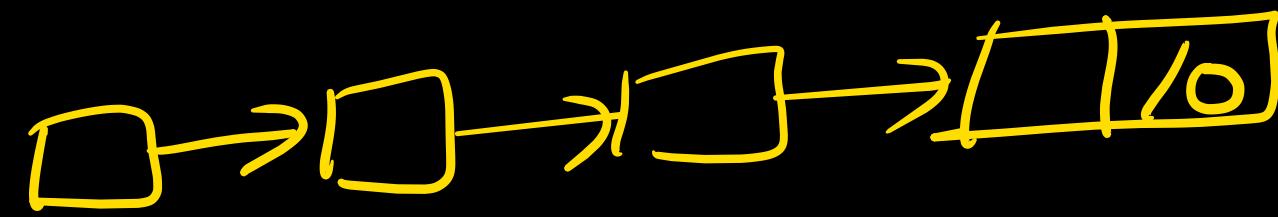


3 compare $S_1 \rightarrow$ data with S_2 data , whichever is smaller,
add that node to the end of resultant LL (R)



Drawbacks of SLL:

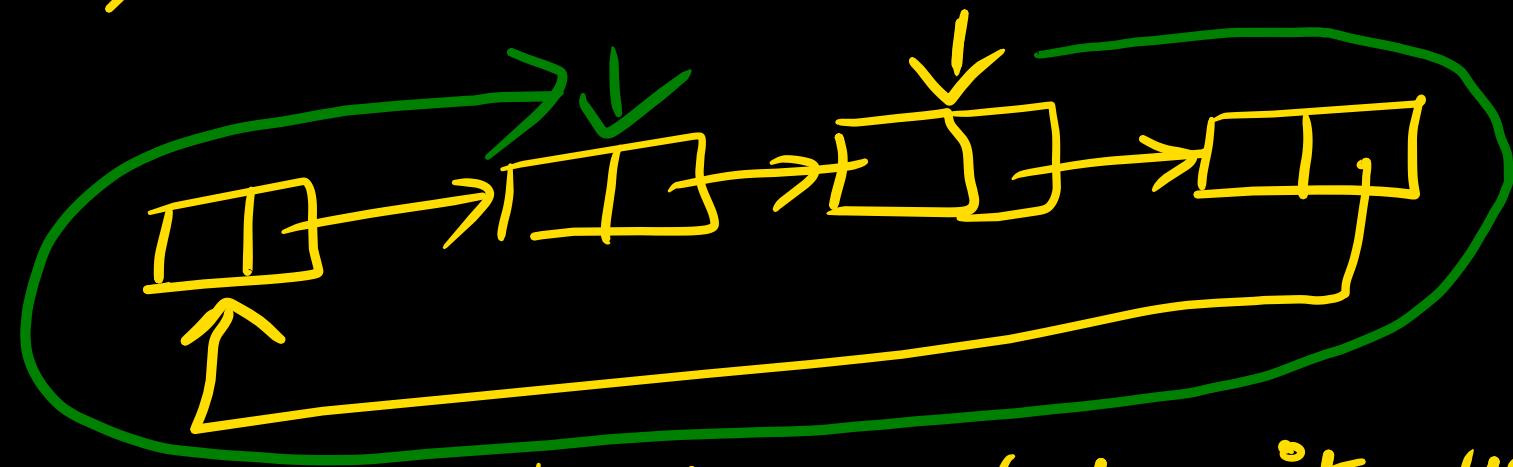
- 1) we can't go back
- 2) last node next pointer is always null.



So to eliminate these 2 drawbacks, we are going to Circular Single linked list.

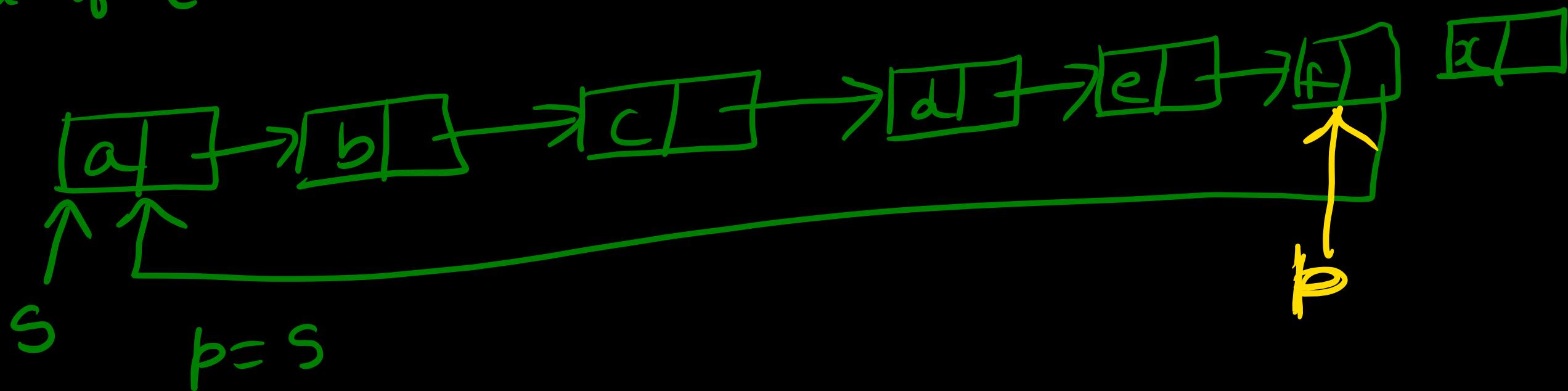
C-SLL :-

- In C-SLL, last node next part is 1st node address.



- we can go back direction, but it will take $O(n)$ time
 $O(n)$ time
- we are utilizing last node next part
efficiently

write an algo to insert a node with data 'x' at the end of C-SLL.

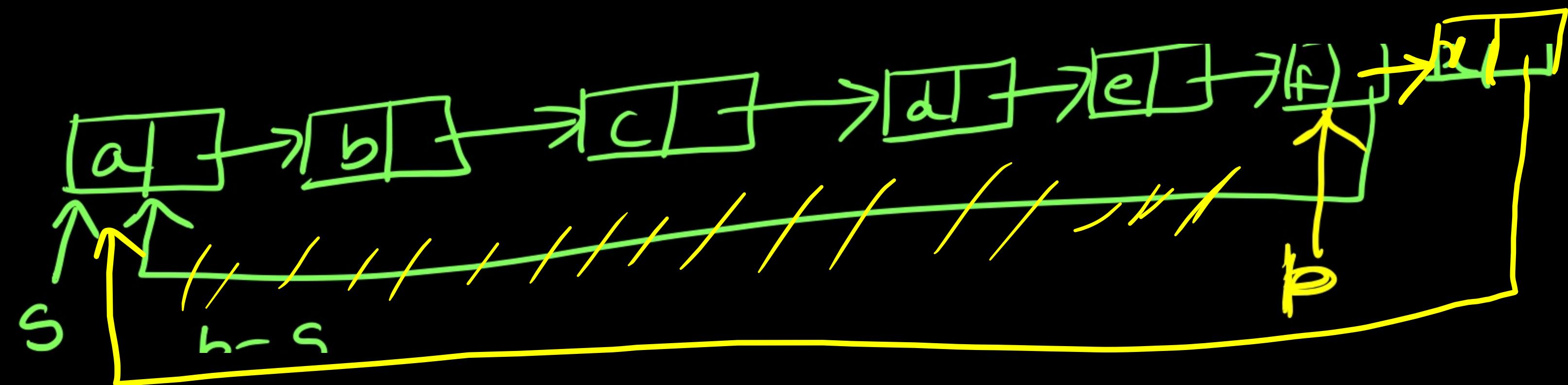


while($b \rightarrow \text{next} \neq s$)

$p = p \rightarrow \text{next}$

$q = (\text{struct node} *) \text{malloc}(\text{size of (struct node)})$;

$q \rightarrow \text{data} = x$



p → next = *q*

q → next = *S*

return (*S*).

Write an algo to add an element of data 'x' at the

beginning of C-SLL



$q = (\text{Struct node}^*) \text{ malloc } (\text{size of (Struct node)})$

$q \rightarrow \text{data} = x.$

while $p = s$

while ($p \rightarrow \text{next} \neq s$)

$p = p \rightarrow \text{next}$

$p \rightarrow \text{next} = q \cdot$

$q \rightarrow \text{next} = s$

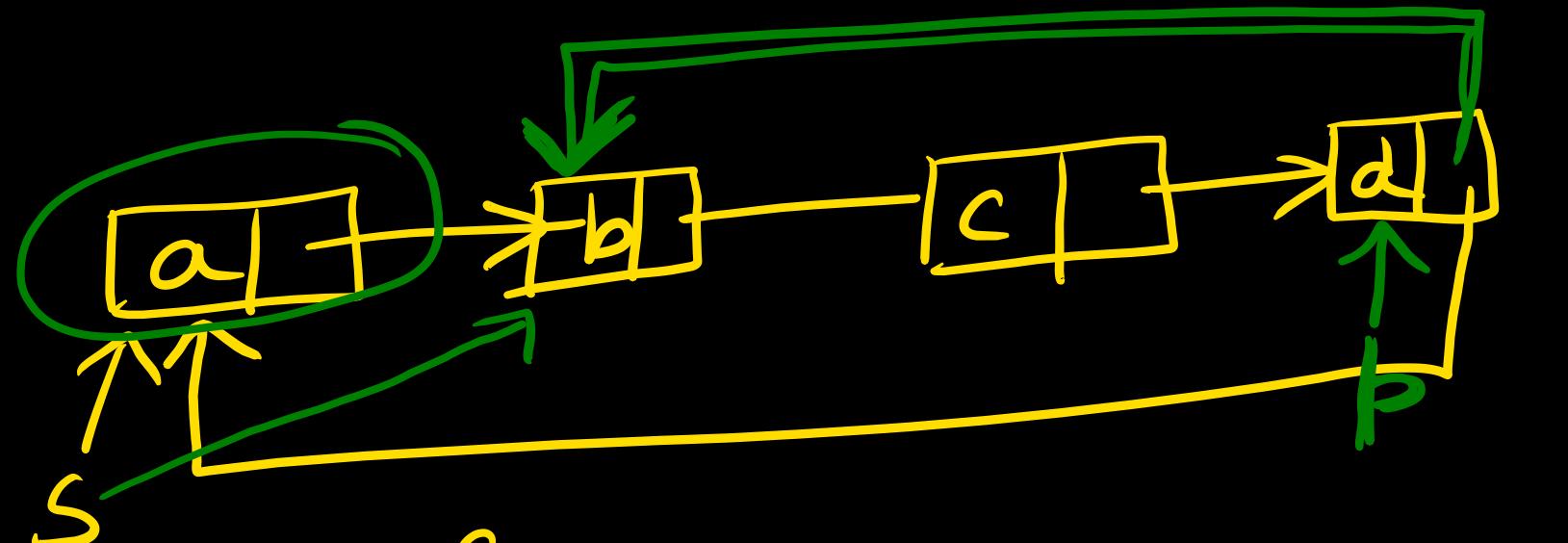
$s = q$

$\text{return}(s).$

edge cases.

- if ($s == \text{null}$)
- if ($s \rightarrow \text{next} == \text{null}$)

gate
gra C SLL delete I node



$p = s$

while ($p \rightarrow \text{next} \neq s$)

① $p = p \rightarrow \text{next};$ → blank

$b \rightarrow \text{next} = s \rightarrow \text{next}$

$\checkmark \text{free}(s); s = p \rightarrow \text{next};$

Corner case

if ($s == \text{null}$)

($s \rightarrow \text{next} != \text{null} == 1$)

(a)

b) \cup d).

Double linked list :-

Struct node

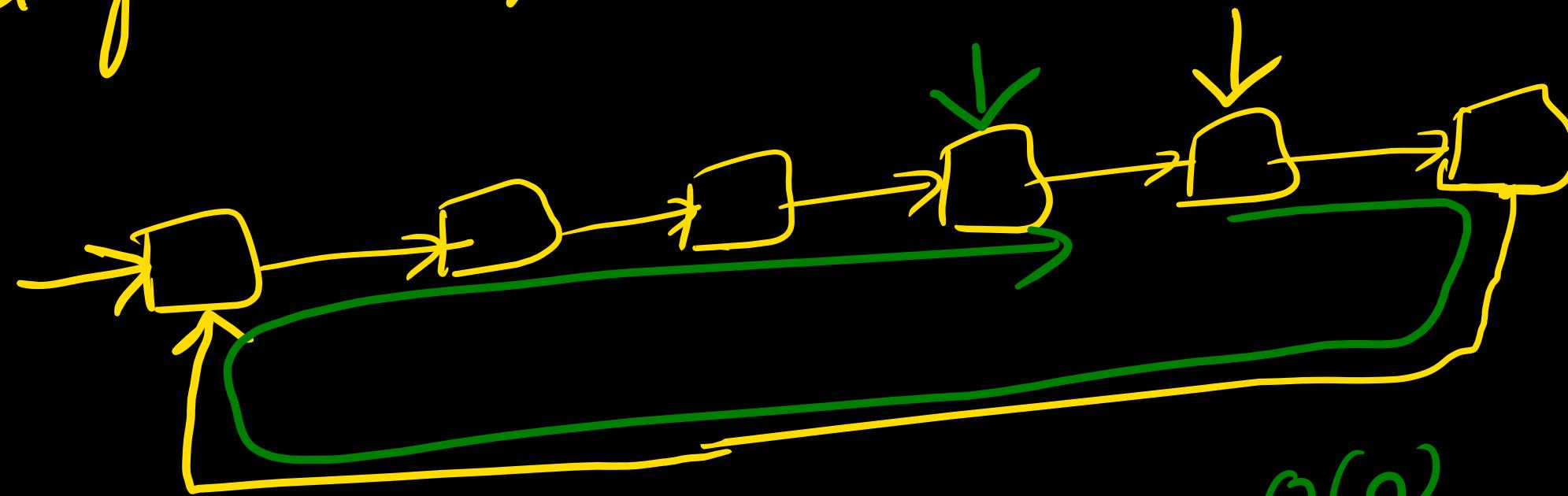
```
{  
    Struct node * prev;  
    int data;  
    Struct node * next;
```

}

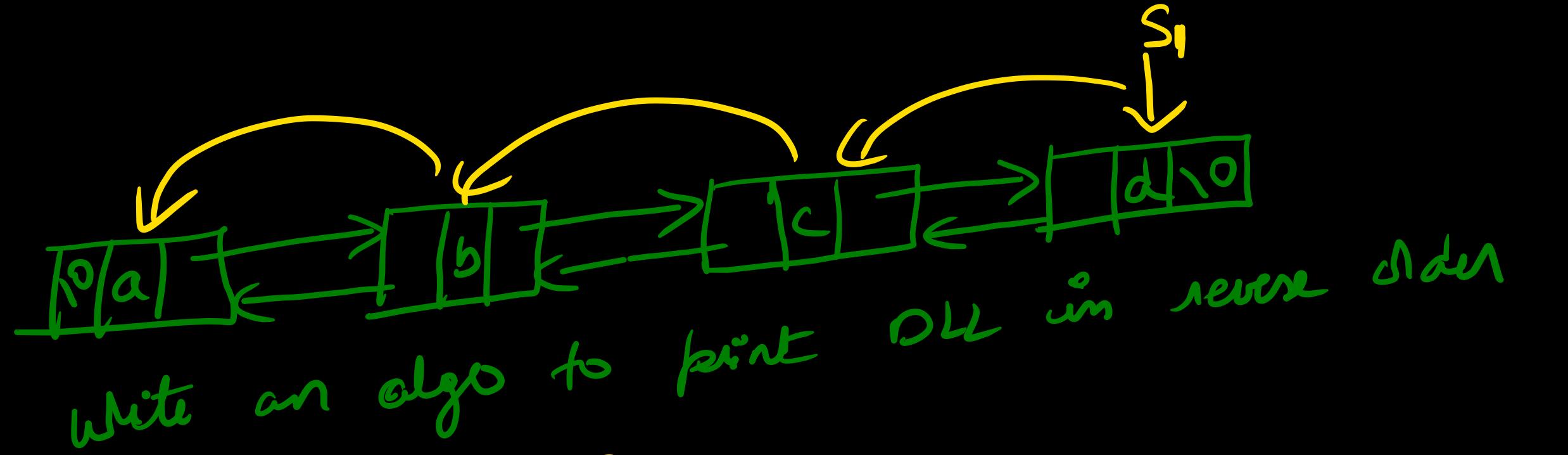


Draw back of CSL :-

Going back required $O(n)$ time



going back seq requires $O(n)$ time

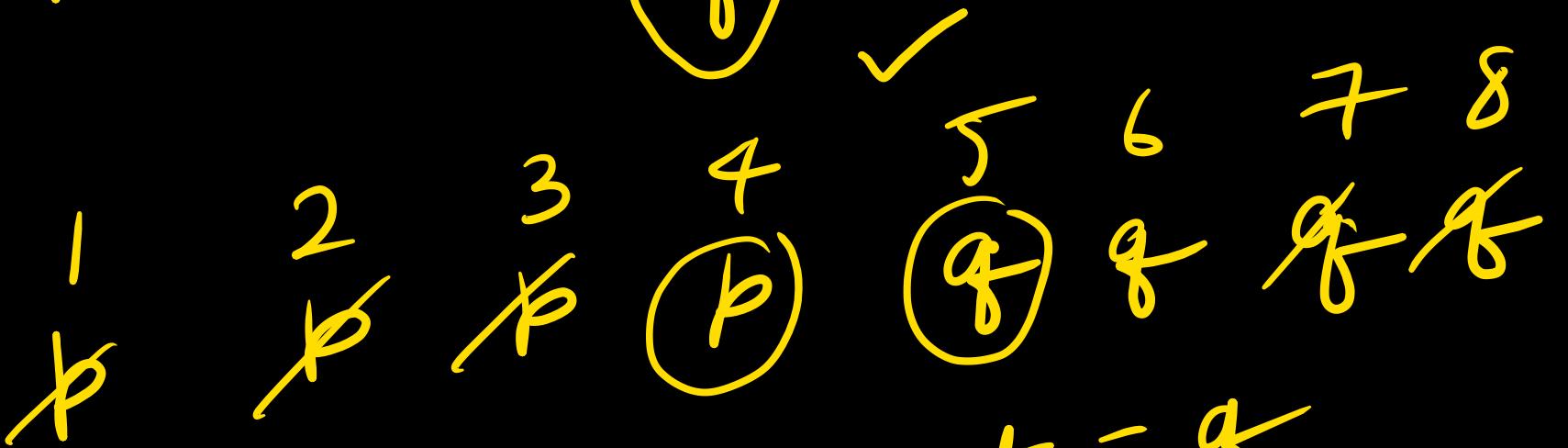
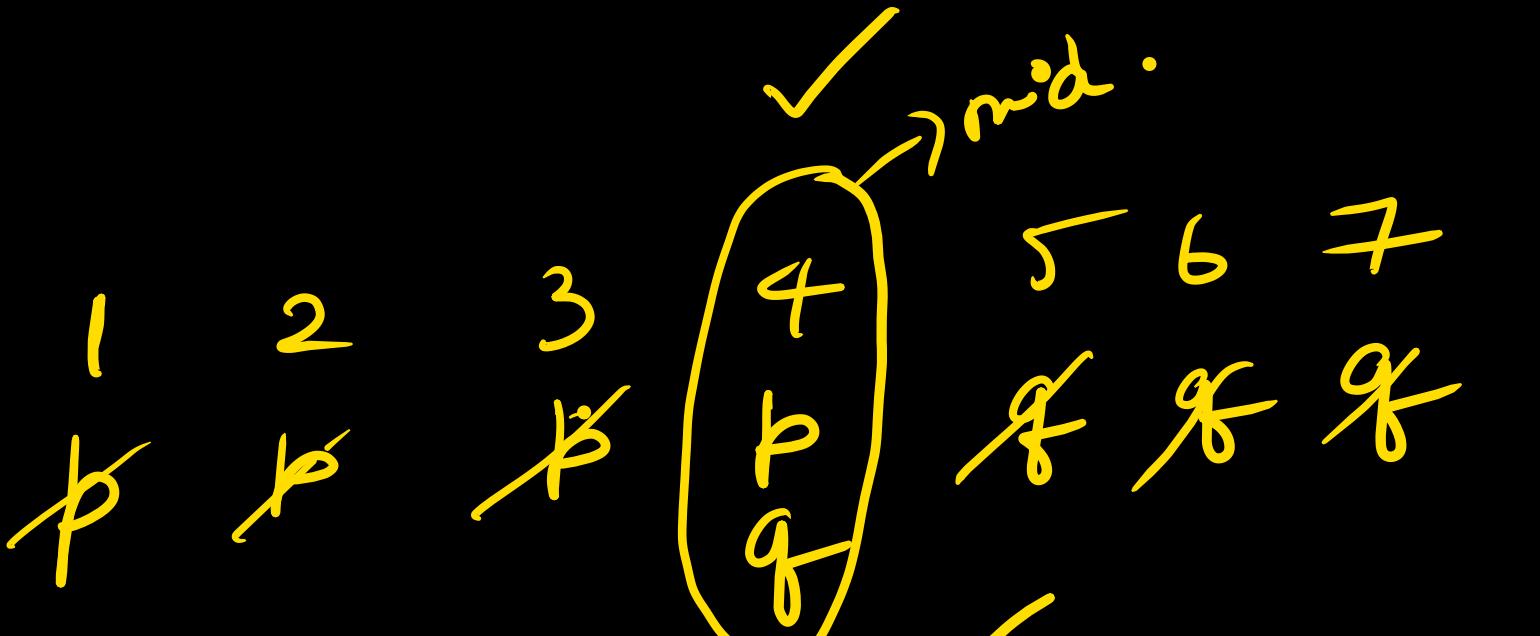


```

 $S_1 = S$ 
while ( $S_1 \rightarrow \text{next} \neq \text{null}$ )
     $S_1 = S_1 \rightarrow \text{next}$ 
    while ( $S_1 \neq \text{null}$ )
        pf ( $S_1 \rightarrow \text{data}$ )
         $S = S_1 \rightarrow \text{prev}$ 
    
```

write an algo to find mid element of DLL.

- 1) use two pointers from the beginning and increment one pointer by one and other by '2'.
- 2) Count the no of nodes and find the middle.
- 3) Start one pointer from beginning and other pointer from end and increment by one step.



$b \rightarrow \text{next} = q$
two mid values.

- ① move q to end of DLL
- ② move p to start of DLL
- ③ while ($p \neq q$ || $p \rightarrow \text{next} = q$)
 - \downarrow odd number
 - \downarrow even

{ $p = p \rightarrow \text{next}$;

$q = q \rightarrow \text{prev}$;

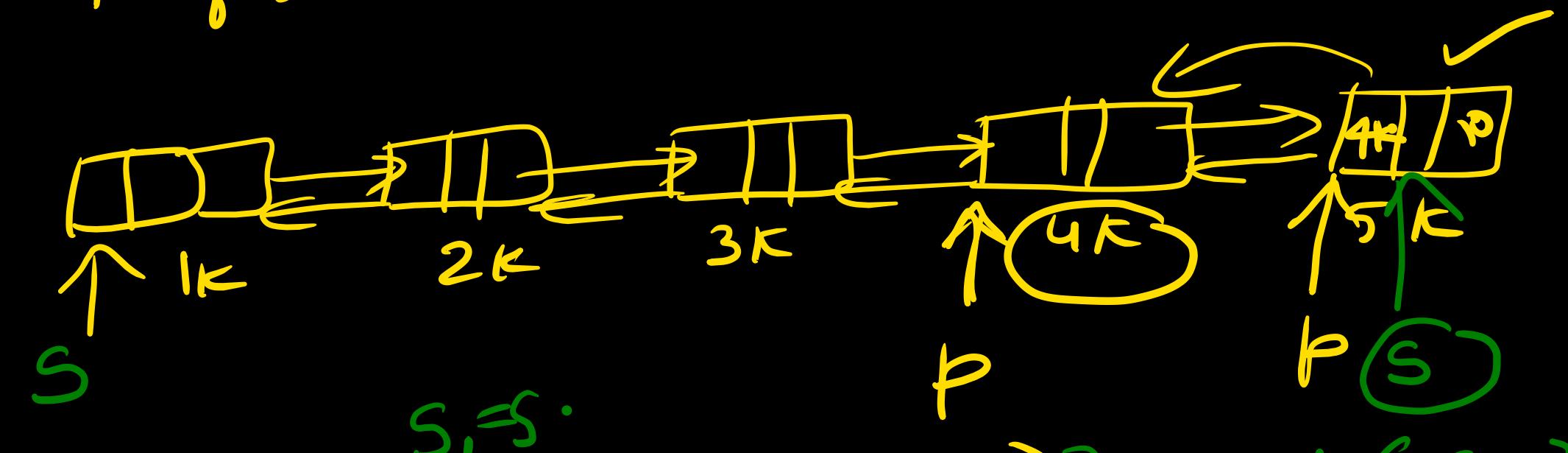
y

Corner cases

~~s~~ $s = \text{null}$

$s \rightarrow \text{next} = \text{null}$

write an algo to find 2nd last node address in DLL



$s_1 = s$.

while (~~$s_1 \rightarrow \text{next} \neq \text{null}$~~ $s_1 \rightarrow \text{next} \neq \text{null}$)

$s_1 = s_1 \rightarrow \text{next}$

Pf ($s_1 \rightarrow \text{prev}$).

while ($s_1 \rightarrow \text{next} \rightarrow \text{next} \neq \text{null}$)

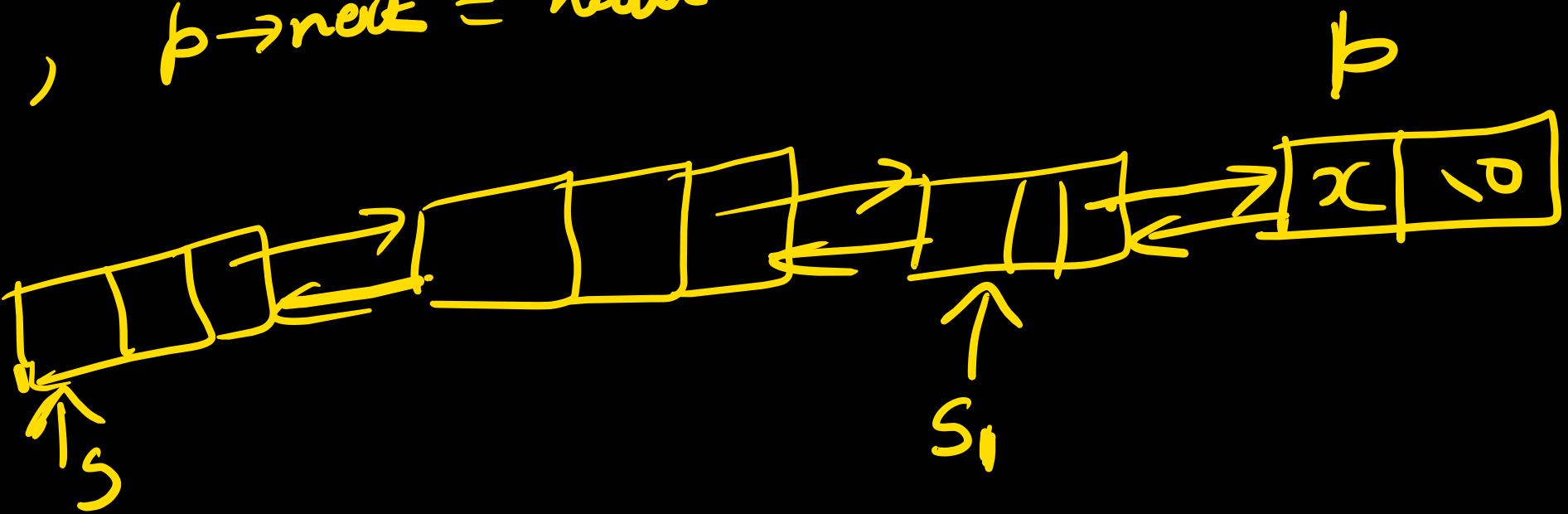
$s_1 = s_1 \rightarrow \text{next}$;

Print(s_1);

Write an algo to add node with data x at end of DLL

$p = (\text{struct node} *) \text{malloc}(\text{size of (struct node)})$

$p \rightarrow \text{data} = x, p \rightarrow \text{next} = \text{null}$



• $S_1 = S$

while ($S_1 \rightarrow \text{next} \neq \text{null}$)

{ $S = S \rightarrow \text{next}$ }

$S_1 \rightarrow \text{next} = p \quad p \rightarrow \text{prev} = S_1$