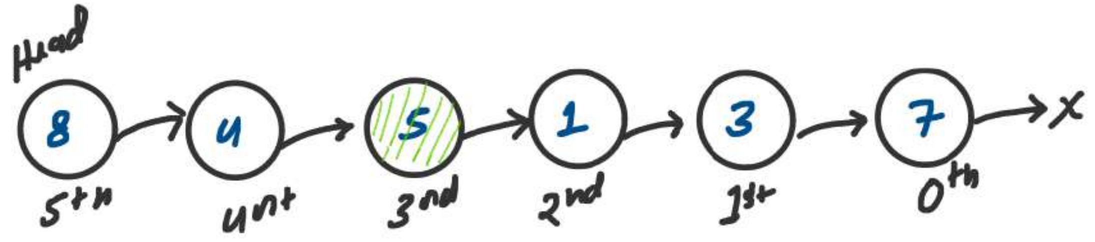
HW 05: Print kth Node from the End (Hacker Rank)

Ex1



pos = kth = 3

output = 5

3rd

RECURSIVE APPROACH

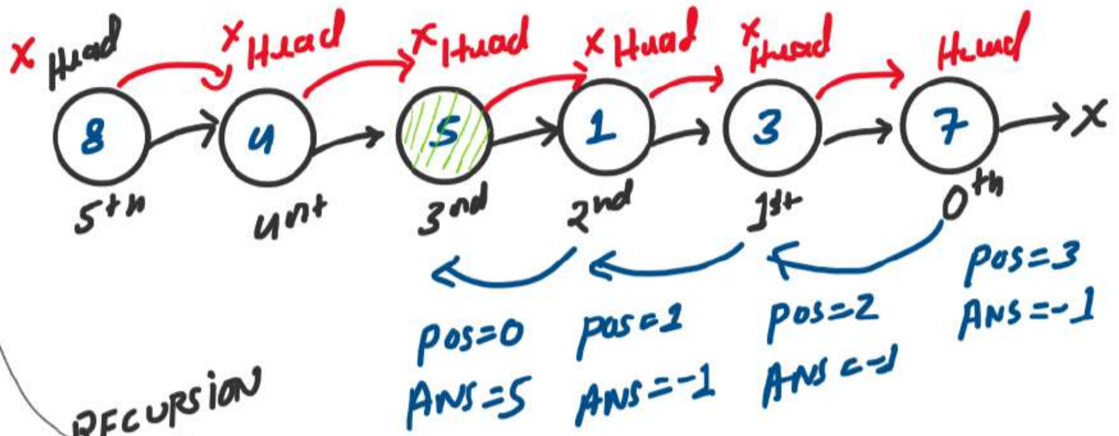
STEP:1 Traverse the list from Head to Tail

STEP:2 Again Traverse the list from Tail to [Jab tak pos==0]

STEP:3 Return Ans = 5

DRY RUN

pos = 3



Step1

```
Base case
  if(!Head)
    return
```

```
fun(Head->next, pos, ANS)
```

RECURSION

Step:3

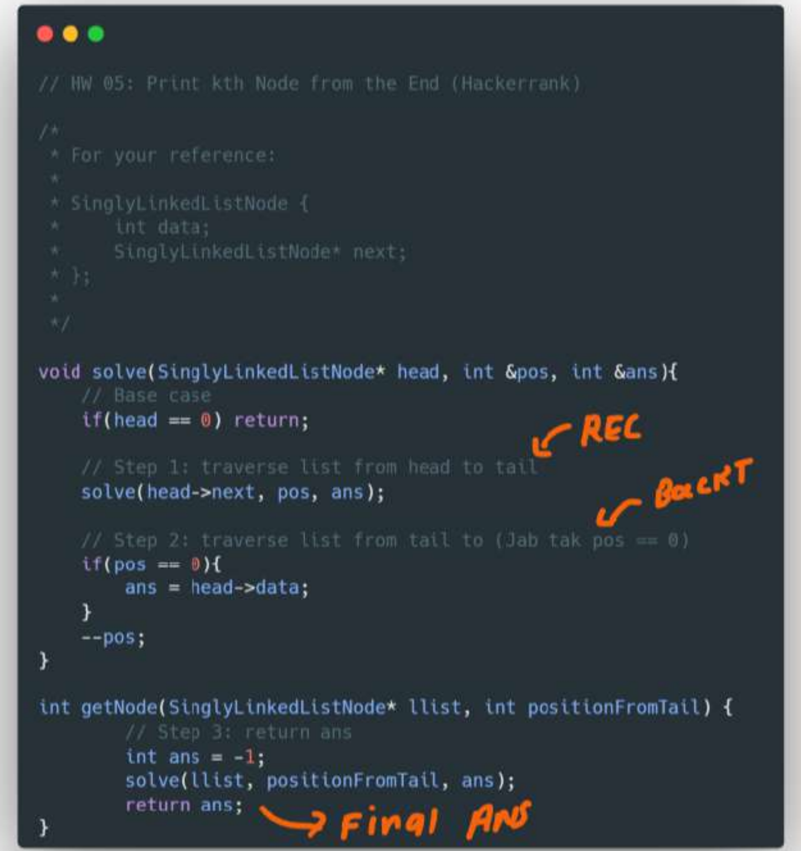
return Ans

Step2

BACKTRACKing

```
if( pos == 0){
    ANS = head->data;
}
pos--;
```

→ catch (Galti ki chances Hai)



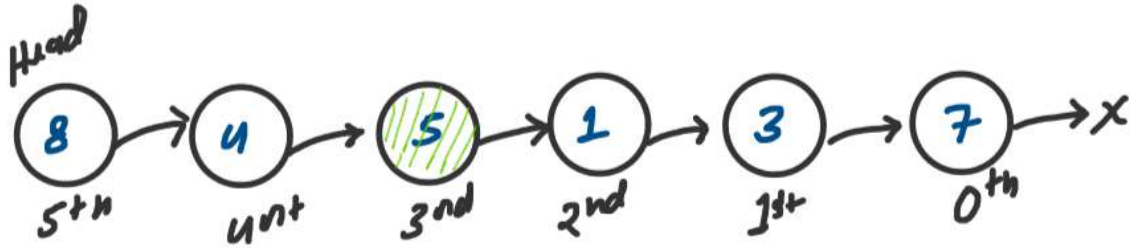
```cpp
// HW 05: Print kth Node from the End (Hackerrank)

/*
 * For your reference:
 *
 * SinglyLinkedListNode {
 *     int data;
 *     SinglyLinkedListNode* next;
 * };
 *
 */

void solve(SinglyLinkedListNode* head, int &pos, int &ans){
    // Base case
    if(head == 0) return;

    // Step 1: traverse list from head to tail
    solve(head->next, pos, ans);

    // Step 2: traverse list from tail to (Jab tak pos == 0)
    if(pos == 0){
        ans = head->data;
    }
    --pos;
}

int getNode(SinglyLinkedListNode* llist, int positionFromTail) {
        // Step 3: return ans
        int ans = -1;
        solve(llist, positionFromTail, ans);
        return ans;
}
```

T.C. ⇒ O(N)

When N is numbers of nodes in the list.

S.C. ⇒ O(1)

Ex1



pos = kth = 3

output =

ITERATIVE APPROACH

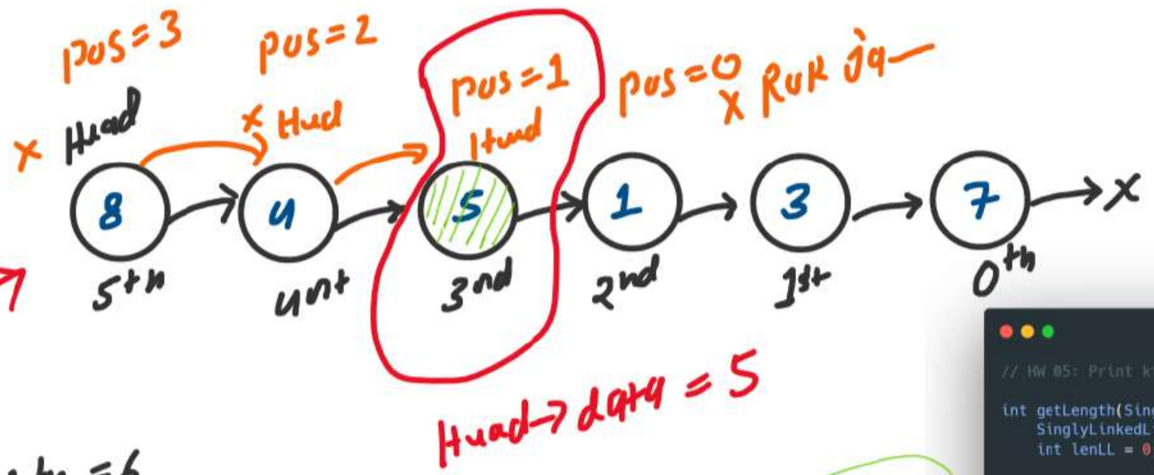
STEP:1 Get length of list

step:2 subtract length - pos.

step:3 Traverse list from Head to [Jab tak pos == 0] .

DRY RUN

K = 3

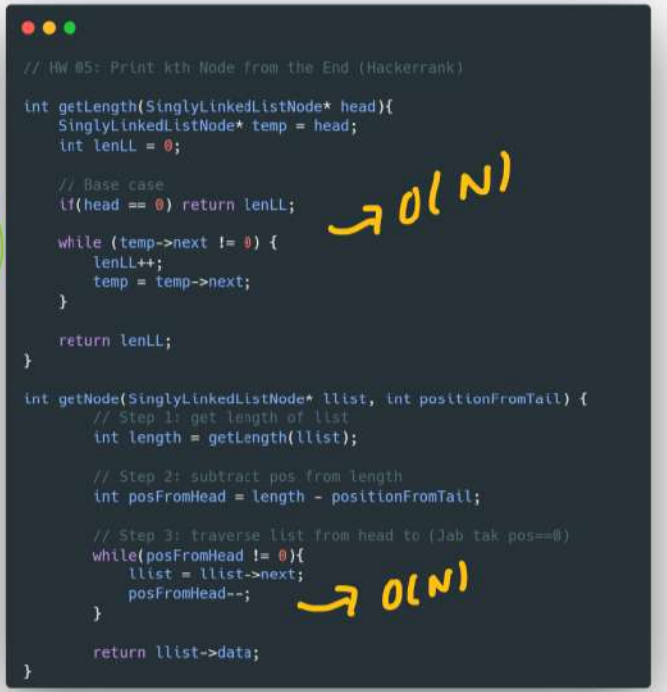


Step:1 length = 6

Step:2 pos = length − K = 6 − 3
= 3

Step:3 return Head→data

$T.C. \Rightarrow O(N) + O(N)$
$= O(N)$
$S.C. \Rightarrow O(1)$



```
// HW 05: Print kth Node from the End (Hackerrank)

int getLength(SinglyLinkedListNode* head){
    SinglyLinkedListNode* temp = head;
    int lenLL = 0;

    // Base case
    if(head == 0) return lenLL;

    while (temp->next != 0) {
        lenLL++;
        temp = temp->next;
    }

    return lenLL;
}

int getNode(SinglyLinkedListNode* llist, int positionFromTail) {
        // Step 1: get length of llist
        int length = getLength(llist);

        // Step 2: subtract pos from length
        int posFromHead = length - positionFromTail;

        // Step 3: traverse list from head to (Jab tak pos==0)
        while(posFromHead != 0){
            llist = llist->next;
            posFromHead--;
        }

        return llist->data;
}
```