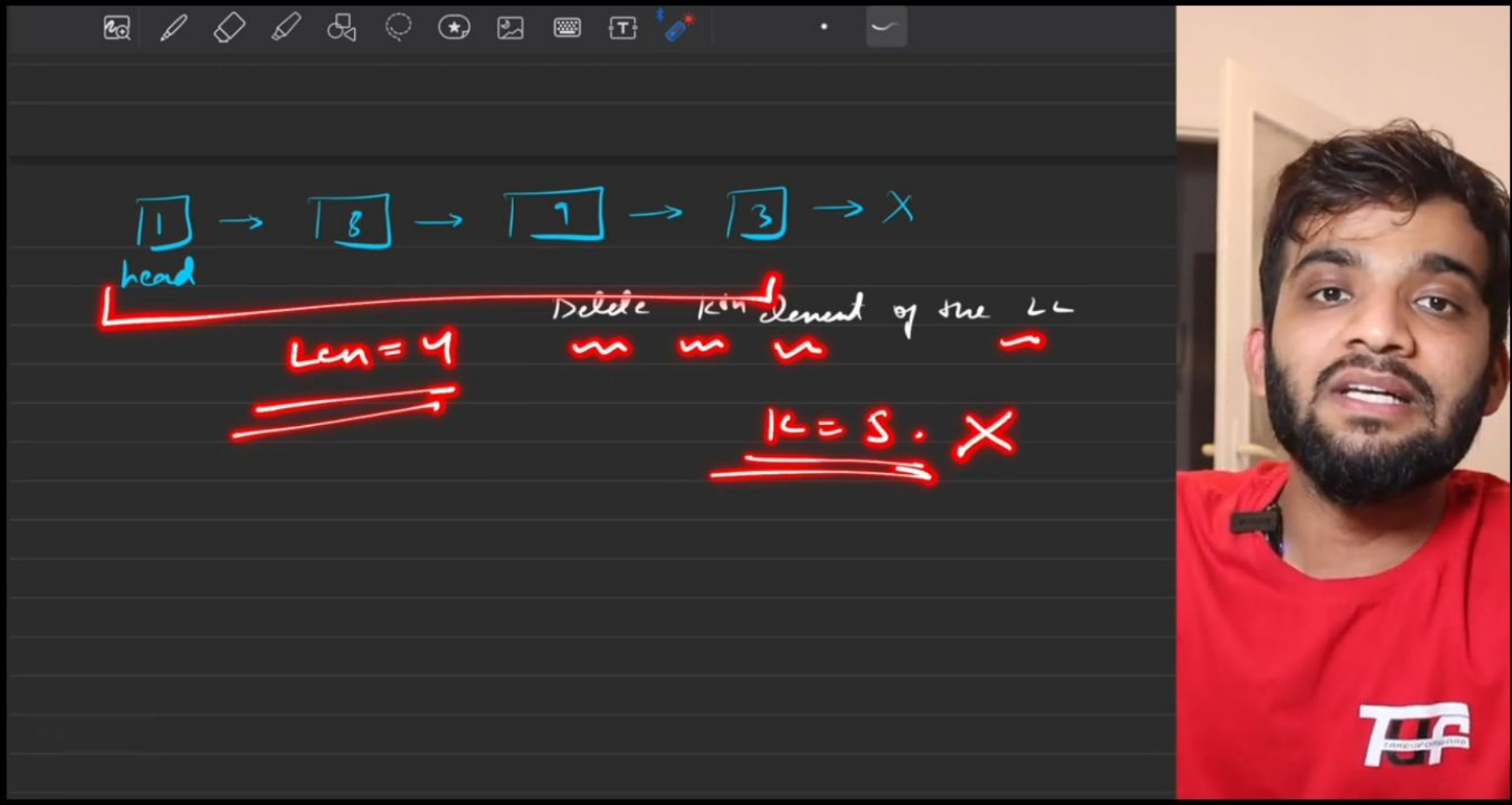
Linked List

- Deletion
- Insertion

→

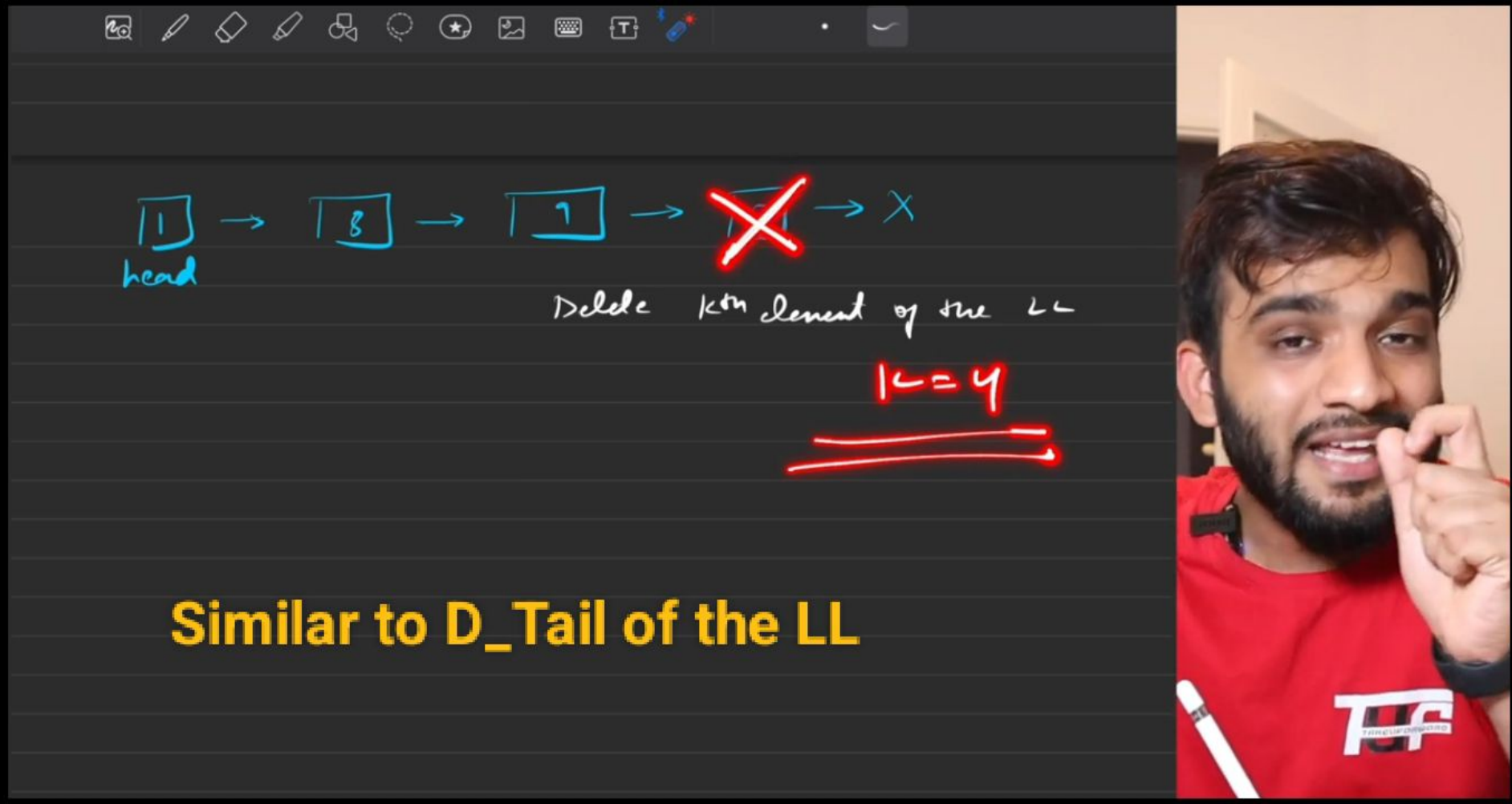
- head ✓
- position → IC
- value
- last ✓

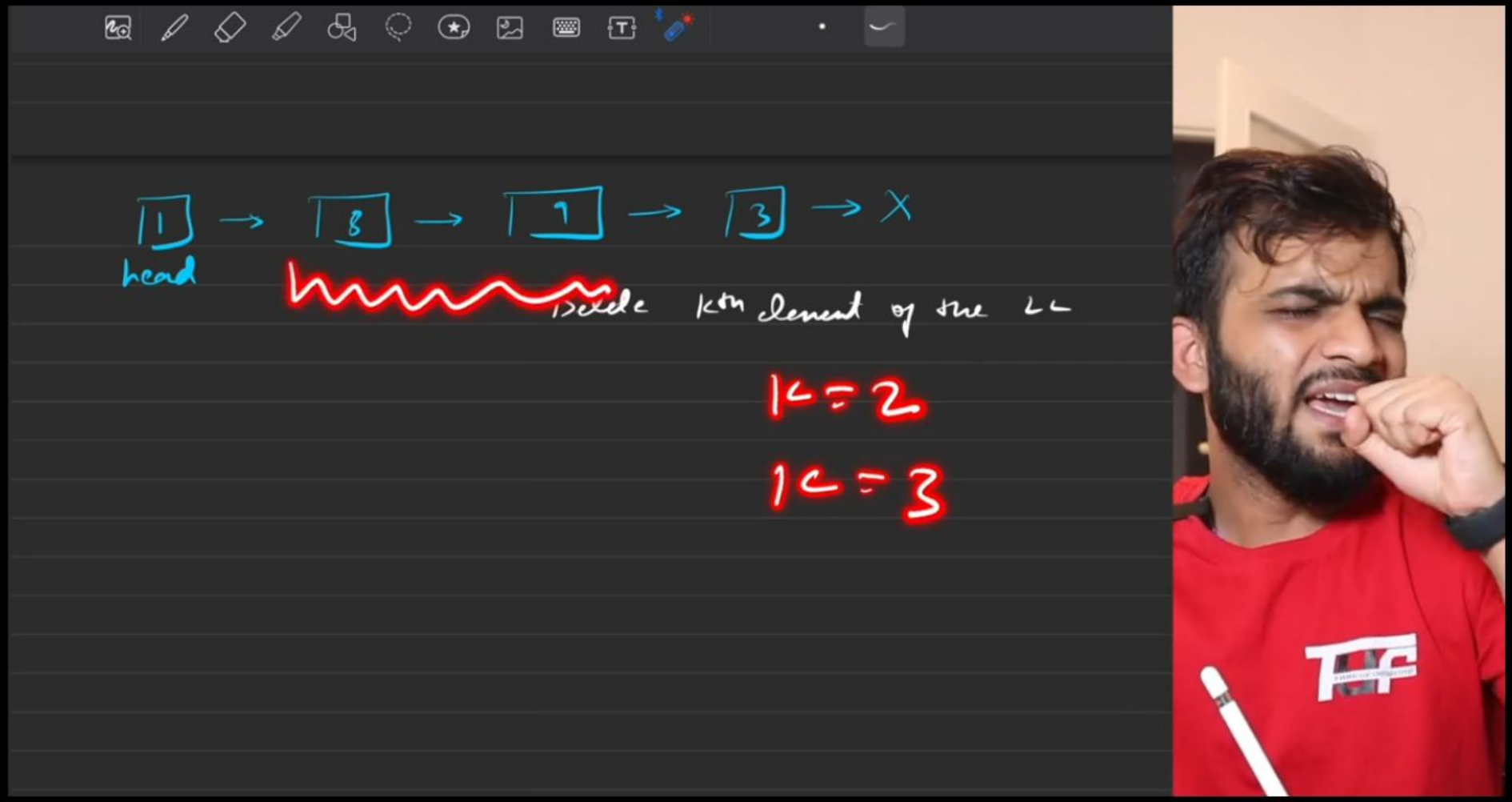

1 → 8 → 7 → 3 → X
head
Delete kth element of the LL
Len = 4
k = 5 . X

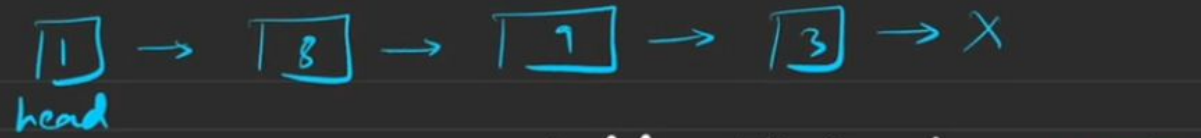
Delete kth element of the LL

k = 1

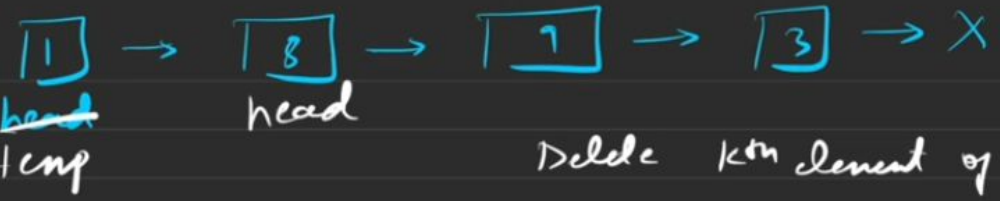
Similar to D_Head of the LL


1
8
7
X
head
Delete kth element of the LL
k=4
Similar to D_Tail of the LL


1
8
7
3
X
head
Delete kth element of the LL
k = 2
k = 3



Delete kth element of



Delete kth element of the LL

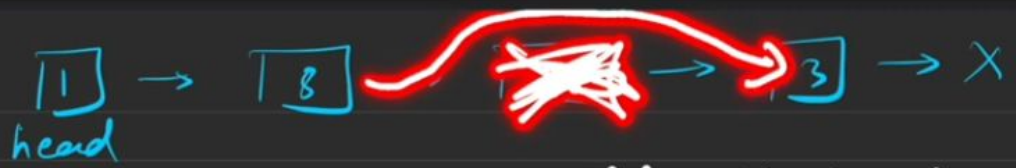
```
Node* deleteK (Node* head, int K)
{
    if (head == NULL) return head;

    if (K == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
```

temp

head

Delete kth element of the LL

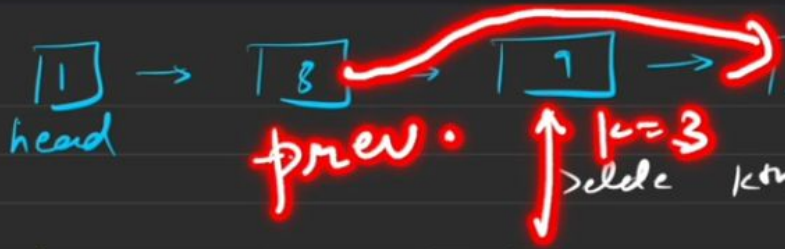
```
Node* deleteK(Node* head, int k)
{
    if (head == NULL) return head;

    if (k == 1) {
        head = head->next;
        return head;
    }
}
```



Delete kth element of the LL

```
Node* deleteK (Node* head, int K)
{
    if (head == NULL) return head;

    if (K == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
```

K = 3




```
Node* deleteK(Node* head, int k)
{
    if (head == NULL) return head;

    if (k == 1) {
        Node* temp = head
        head = head->next;
        free (temp)
        return head;
    }
```

k = 3

prev->next = prev->next->next

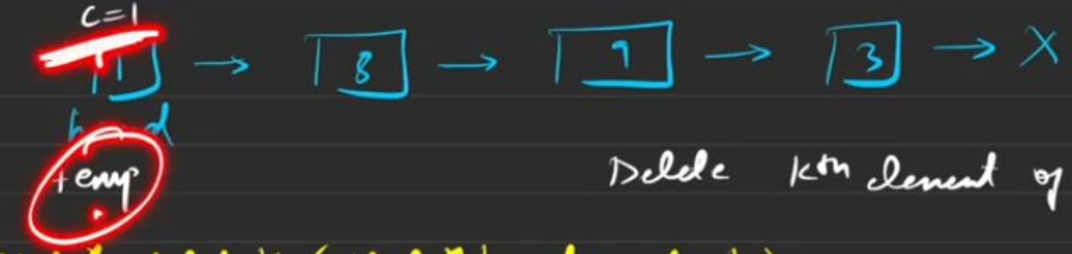
Delete kth element of the LL

```
Node* deleteK(Node* head, int k)
{
    if (head == NULL) return head;

    if (k == 1) {
        Node* temp = head
        head = head->next;
        free (temp)
        return head;
    }
```

k = 3

k = 5

```
while (temp != null)
{
    cnt++;
    if (cnt == k)
    {
        prev->next = prev->next->next;
        break;
    }
    temp = temp->next;
}
```



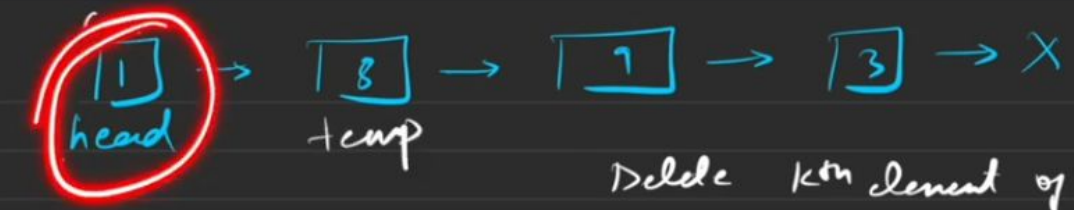
Delete kth element of the LL

```
Node* deleteK (Node* head, int K)
{
    if (head == NULL) return head;

    if (K == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
}
```

K = 2

K = 5

cnt = 0 , Node* temp = head; prev = null;



Delete kth element of the LL

```
Node* deleteK(Node* head, int k)
{
    if (head == null) return head;

    if (k == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
    cnt = 0, Node* temp = head; prev = null;
```

k = 3

```
        cnt++;
        if (cnt == k)
        {
            prev->next = prev->next->next;
            break;
        }
    }
    prev = temp
    temp = temp->next;
}
```

Delete kth element of the LL

```
Node* deleteK(Node* head, int k)
{
    if (head == null) return head;

    if (k == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
    cnt = 0, Node* temp = head; prev = null;

    while (temp != null)
```

k = 3

Delete kth element of the LL

```
Node* deleteK (Node* head, int k)
{
    if (head == null) return head;

    if (k == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
    cnt = 0 , Node* temp = head; prev = null;
```

k = 3

Delete kth element of the LL

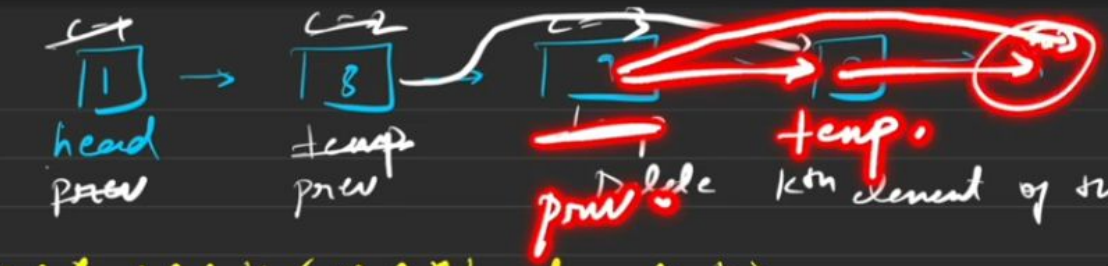
```
Node* deleteK(Node* head, int k)                    k = 3
{
    if (head == null) return head;

    if (k == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
    cnt = 0, Node* temp = head; prev = null;
```

Delete kth element of the LL

```
Node* deleteK(Node* head, int k)
{
    if (head == null) return head;

    if (k == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
    cnt = 0, Node* temp = head; prev = null;

    while (temp != null)
```

k = 3

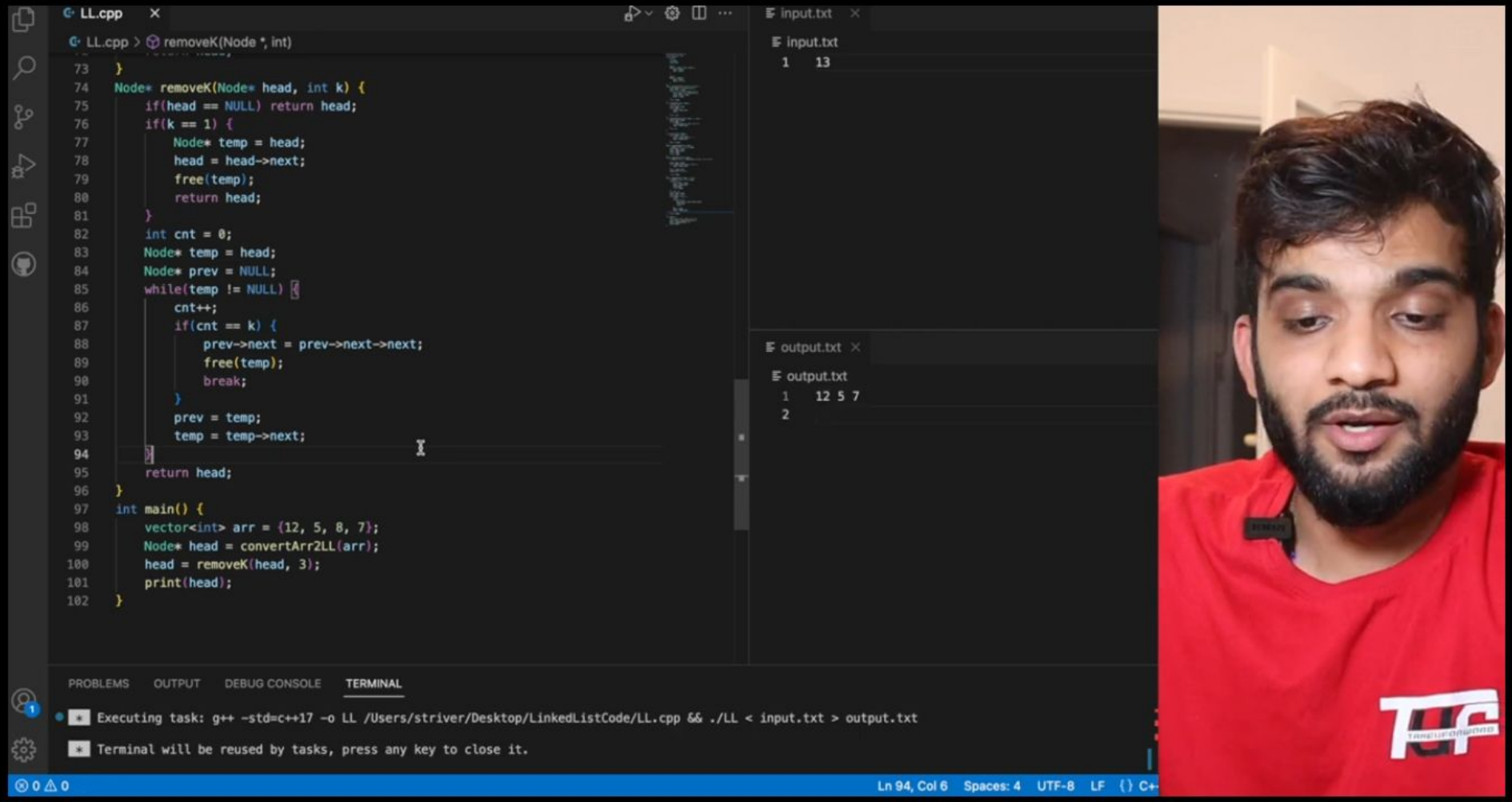
```
if (cnt == k)
{
    prev->next = prev->next->next;
    break; free(temp) break;
}
prev = temp
temp = temp->next;
}
return head;
}
```

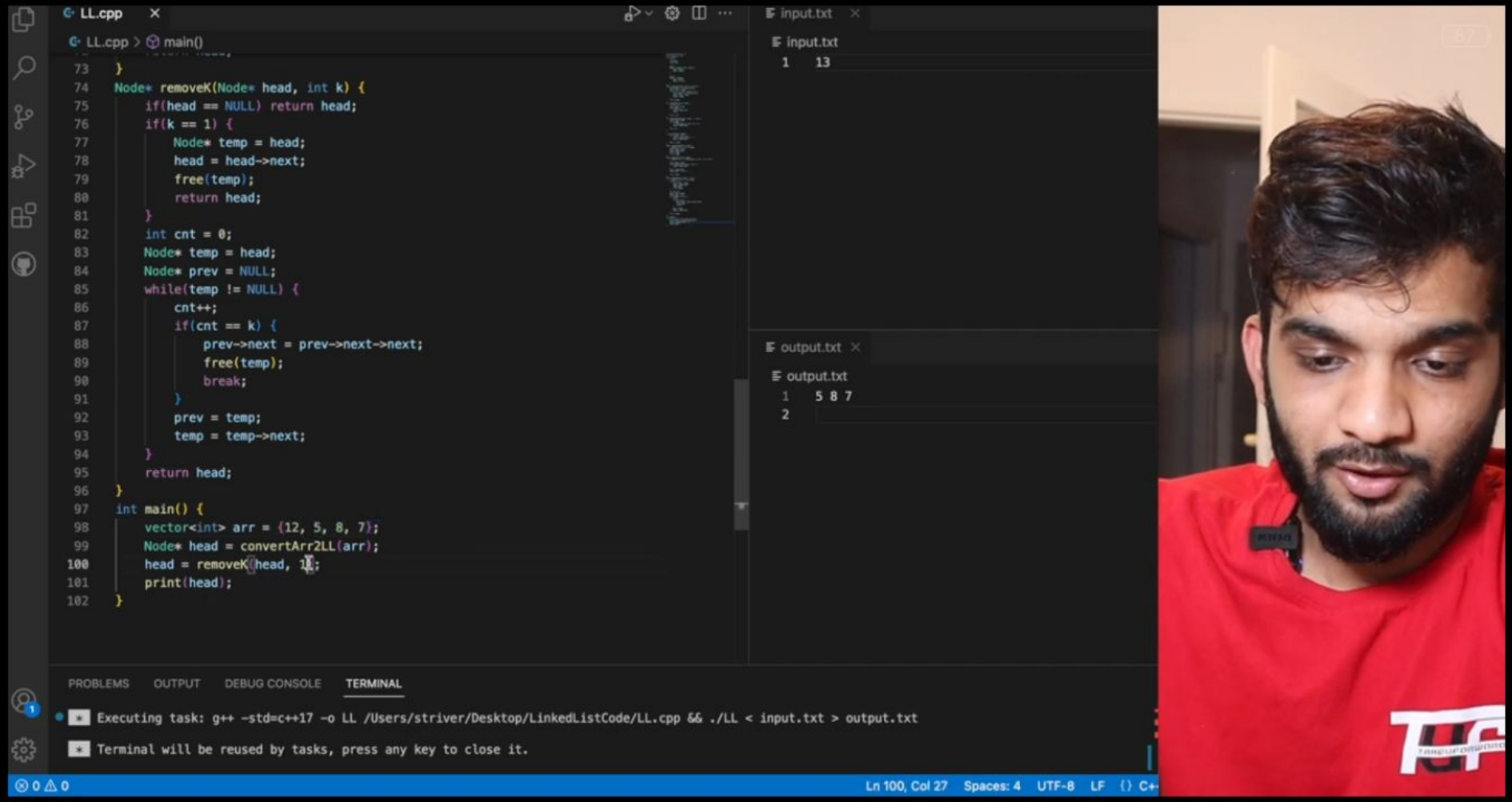



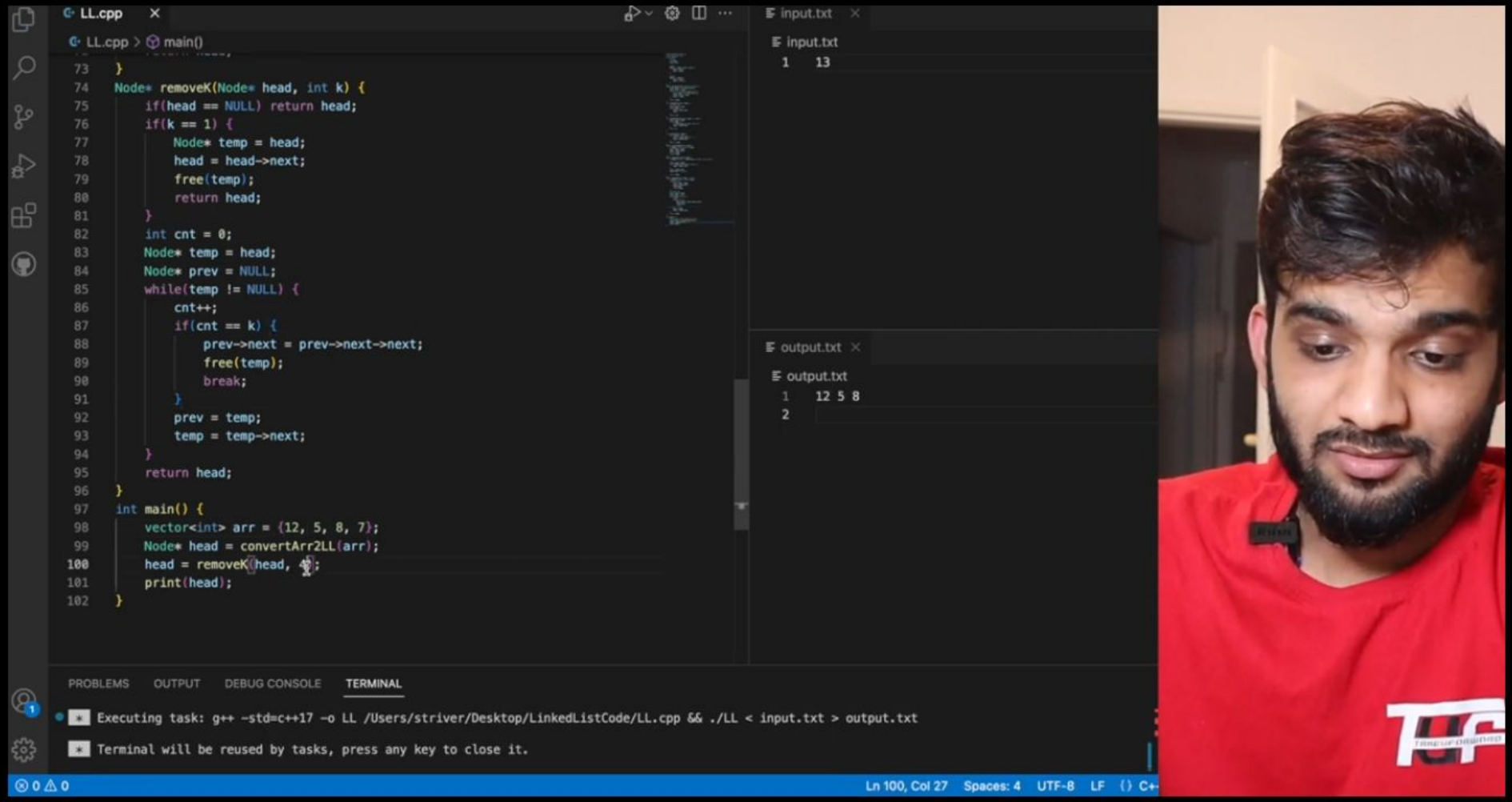
Delete kth element of the LL

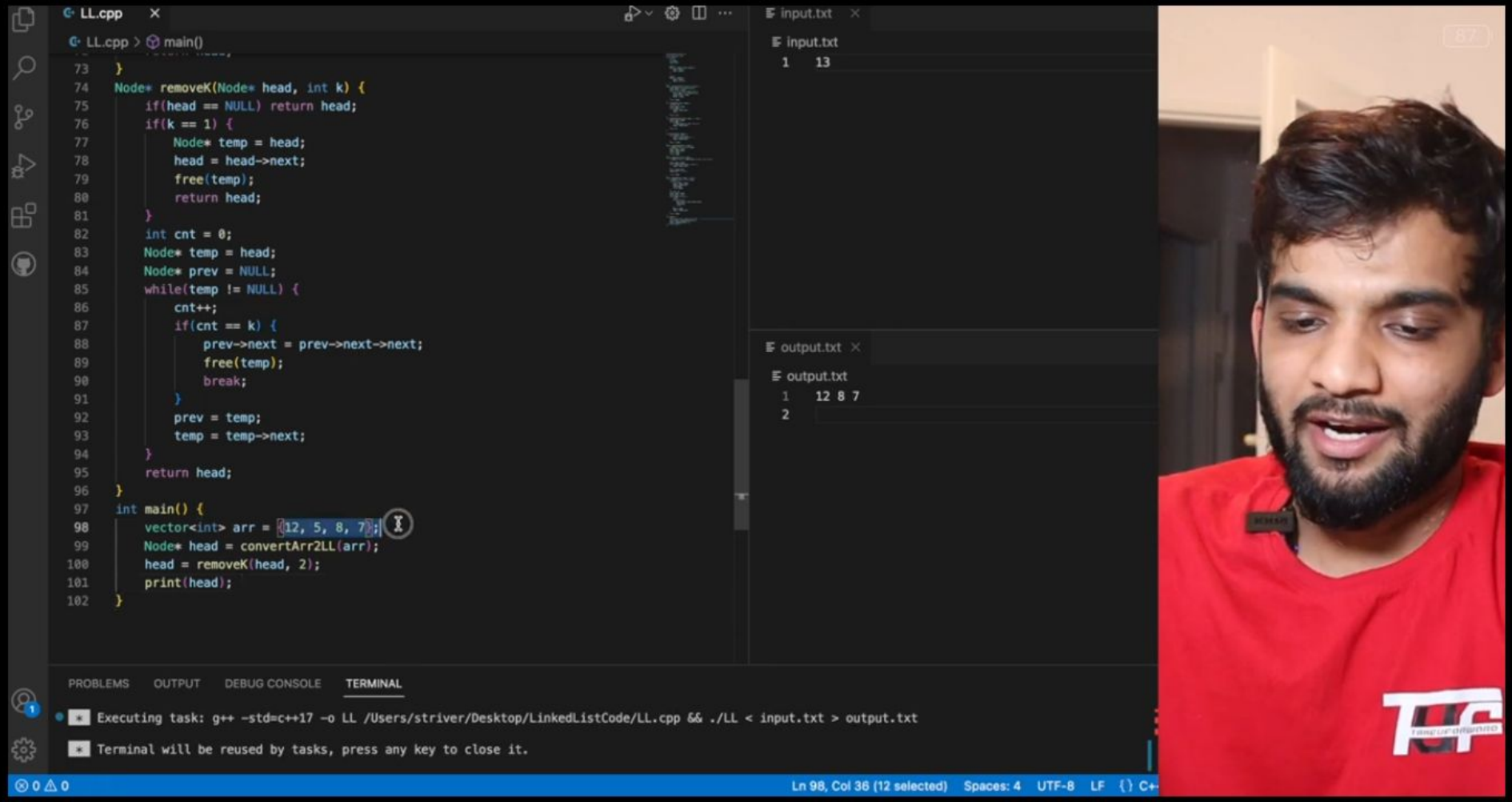
```
Node* deleteK (Node* head, int k)
{
    if (head == NULL) return head;

    if (k == 1) {
        Node* temp = head
        head = head -> next;
        free (temp)
        return head;
    }
    cnt = 0 , Node* temp = head; prev = NULL;
```

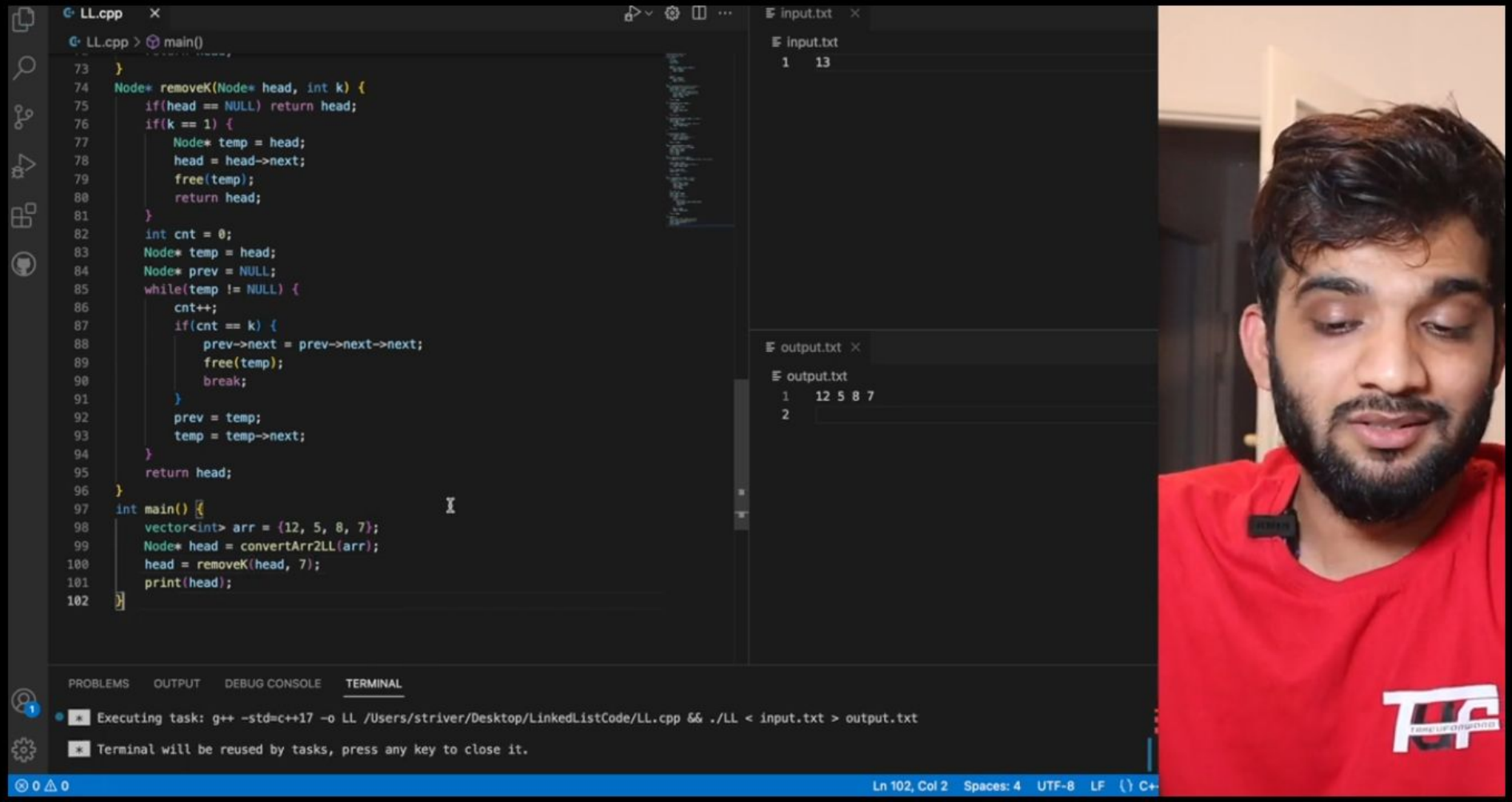
k = 4


LL.cpp
LL.cpp > removeK(Node *, int)
}
Node* removeK(Node* head, int k) {
if(head == NULL) return head;
if(k == 1) {
Node* temp = head;
head = head->next;
free(temp);
return head;
}
int cnt = 0;
Node* temp = head;
Node* prev = NULL;
while(temp != NULL) {
cnt++;
if(cnt == k) {
prev->next = prev->next->next;
free(temp);
break;
}
prev = temp;
temp = temp->next;
}
return head;
}
int main() {
vector<int> arr = {12, 5, 8, 7};
Node* head = convertArr2LL(arr);
head = removeK(head, 3);
print(head);
}
input.txt
input.txt
1 13
output.txt
output.txt
1 12 5 7
2
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Executing task: g++ -std=c++17 -o LL /Users/striver/Desktop/LinkedListCode/LL.cpp && ./LL < input.txt > output.txt
Terminal will be reused by tasks, press any key to close it.
Ln 94, Col 6
Spaces: 4
UTF-8
LF


LL.cpp
LL.cpp > main()
}
Node* removeK(Node* head, int k) {
if(head == NULL) return head;
if(k == 1) {
Node* temp = head;
head = head->next;
free(temp);
return head;
}
int cnt = 0;
Node* temp = head;
Node* prev = NULL;
while(temp != NULL) {
cnt++;
if(cnt == k) {
prev->next = prev->next->next;
free(temp);
break;
}
prev = temp;
temp = temp->next;
}
return head;
}
int main() {
vector<int> arr = {12, 5, 8, 7};
Node* head = convertArr2LL(arr);
head = removeK(head, 1);
print(head);
}
input.txt
input.txt
1 13
output.txt
output.txt
1 5 8 7
2
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Executing task: g++ -std=c++17 -o LL /Users/striver/Desktop/LinkedListCode/LL.cpp && ./LL < input.txt > output.txt
Terminal will be reused by tasks, press any key to close it.
Ln 100, Col 27
Spaces: 4
UTF-8
LF


LL.cpp
LL.cpp > main()
}
Node* removeK(Node* head, int k) {
if(head == NULL) return head;
if(k == 1) {
Node* temp = head;
head = head->next;
free(temp);
return head;
}
int cnt = 0;
Node* temp = head;
Node* prev = NULL;
while(temp != NULL) {
cnt++;
if(cnt == k) {
prev->next = prev->next->next;
free(temp);
break;
}
prev = temp;
temp = temp->next;
}
return head;
}
int main() {
vector<int> arr = {12, 5, 8, 7};
Node* head = convertArr2LL(arr);
head = removeK(head, 4);
print(head);
}
input.txt
input.txt
1 13
output.txt
output.txt
1 12 5 8
2
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Executing task: g++ -std=c++17 -o LL /Users/striver/Desktop/LinkedListCode/LL.cpp && ./LL < input.txt > output.txt
Terminal will be reused by tasks, press any key to close it.
Ln 100, Col 27
Spaces: 4
UTF-8
LF


LL.cpp
LL.cpp > main()
}
Node* removeK(Node* head, int k) {
if(head == NULL) return head;
if(k == 1) {
Node* temp = head;
head = head->next;
free(temp);
return head;
}
int cnt = 0;
Node* temp = head;
Node* prev = NULL;
while(temp != NULL) {
cnt++;
if(cnt == k) {
prev->next = prev->next->next;
free(temp);
break;
}
prev = temp;
temp = temp->next;
}
return head;
}
int main() {
vector<int> arr = {12, 5, 8, 7};
Node* head = convertArr2LL(arr);
head = removeK(head, 2);
print(head);
}
input.txt
input.txt
1 13
output.txt
output.txt
1 12 8 7
2
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Executing task: g++ -std=c++17 -o LL /Users/striver/Desktop/LinkedListCode/LL.cpp && ./LL < input.txt > output.txt
Terminal will be reused by tasks, press any key to close it.
Ln 98, Col 36 (12 selected)
Spaces: 4
UTF-8
LF


LL.cpp
LL.cpp > main()
}
Node* removeK(Node* head, int k) {
if(head == NULL) return head;
if(k == 1) {
Node* temp = head;
head = head->next;
free(temp);
return head;
}
int cnt = 0;
Node* temp = head;
Node* prev = NULL;
while(temp != NULL) {
cnt++;
if(cnt == k) {
prev->next = prev->next->next;
free(temp);
break;
}
prev = temp;
temp = temp->next;
}
return head;
}
int main() {
vector<int> arr = {12, 5, 8, 7};
Node* head = convertArr2LL(arr);
head = removeK(head, 7);
print(head);
}
input.txt
input.txt
1 13
output.txt
output.txt
1 12 5 8 7
2
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Executing task: g++ -std=c++17 -o LL /Users/striver/Desktop/LinkedListCode/LL.cpp && ./LL < input.txt > output.txt
Terminal will be reused by tasks, press any key to close it.
Ln 102, Col 2
Spaces: 4
UTF-8
LF