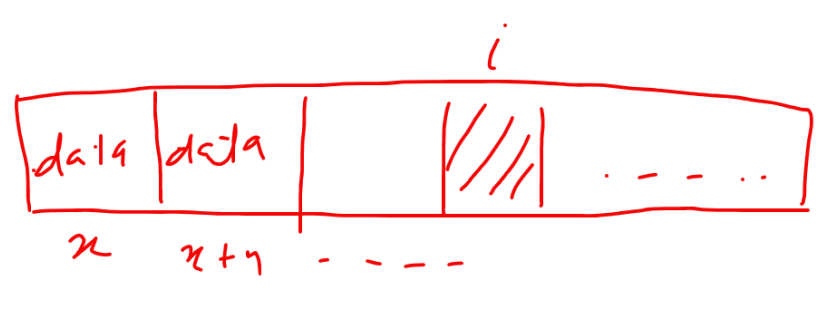
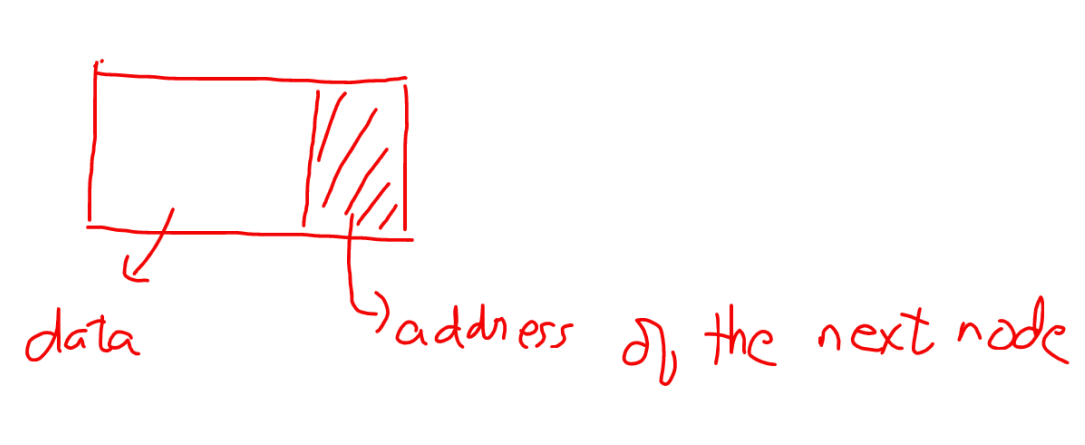
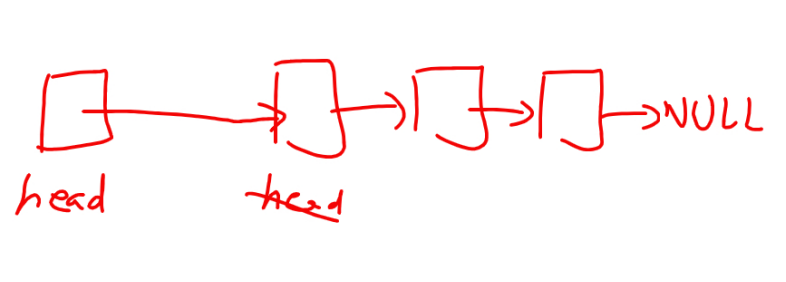
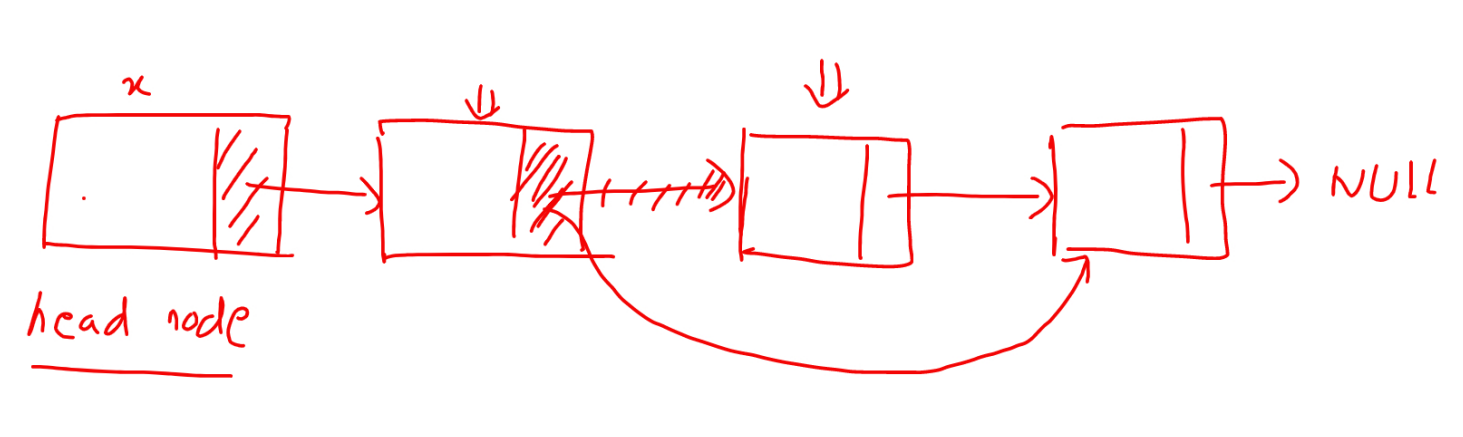
Arrays:


i
data
data
x
x+1

In L.L., elements are not stored in contiguous memory locations (unlike arrays).


data
address of the next node


head
~~head~~
NULL


x
head node
NULL

Adv. of L.L over arrays:

1) No upper limit of size unlike static arrays.
2) Insertion at the beginning of a L.L. takes O(1) time.
3) No shifting of elements is required to delete something from a L.L.

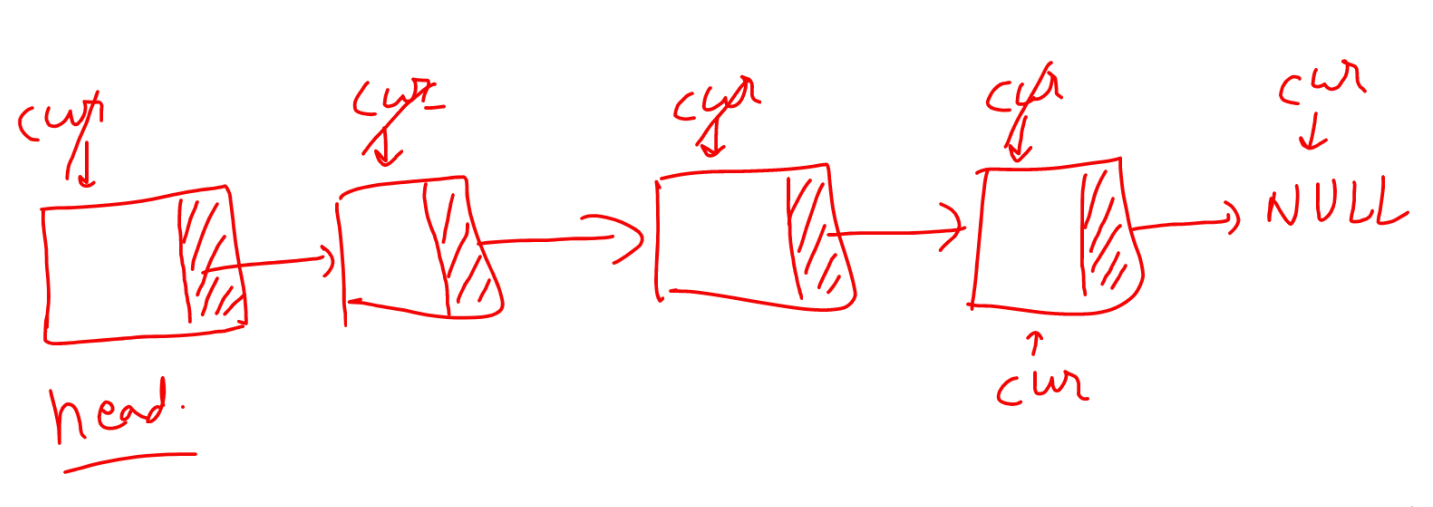
Disadv. of L.L:

1) No contiguous memory allocation leads to no random access. Any element can be accessed only in O(n) time.
2) Requires some extra storage compared to arrays.
3) Not cache-friendly.

C++
x→data
x→next

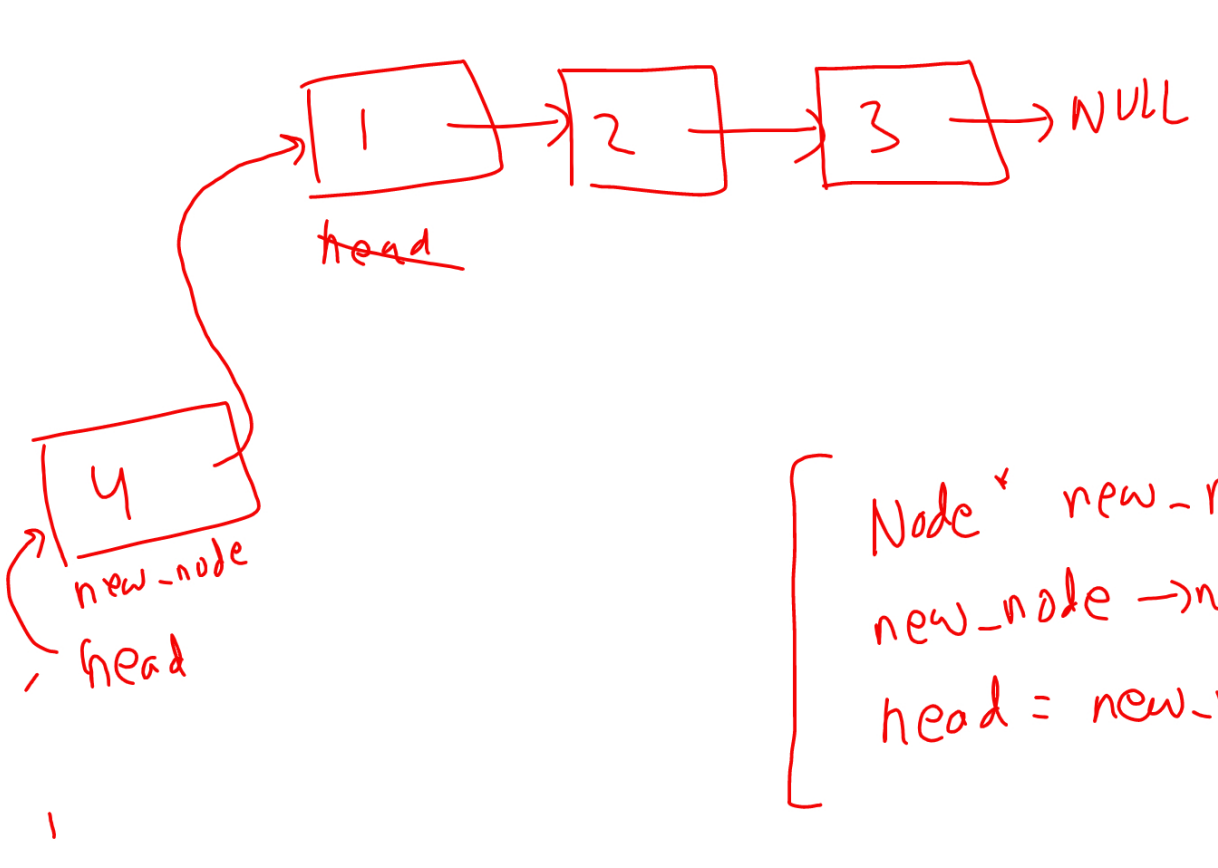
Java.
x.data
x.next.

```
class Node {
   public:
      int data;
      Node* next;
};
```


cur
cur
cur
cur
cur
NULL
head.
cur

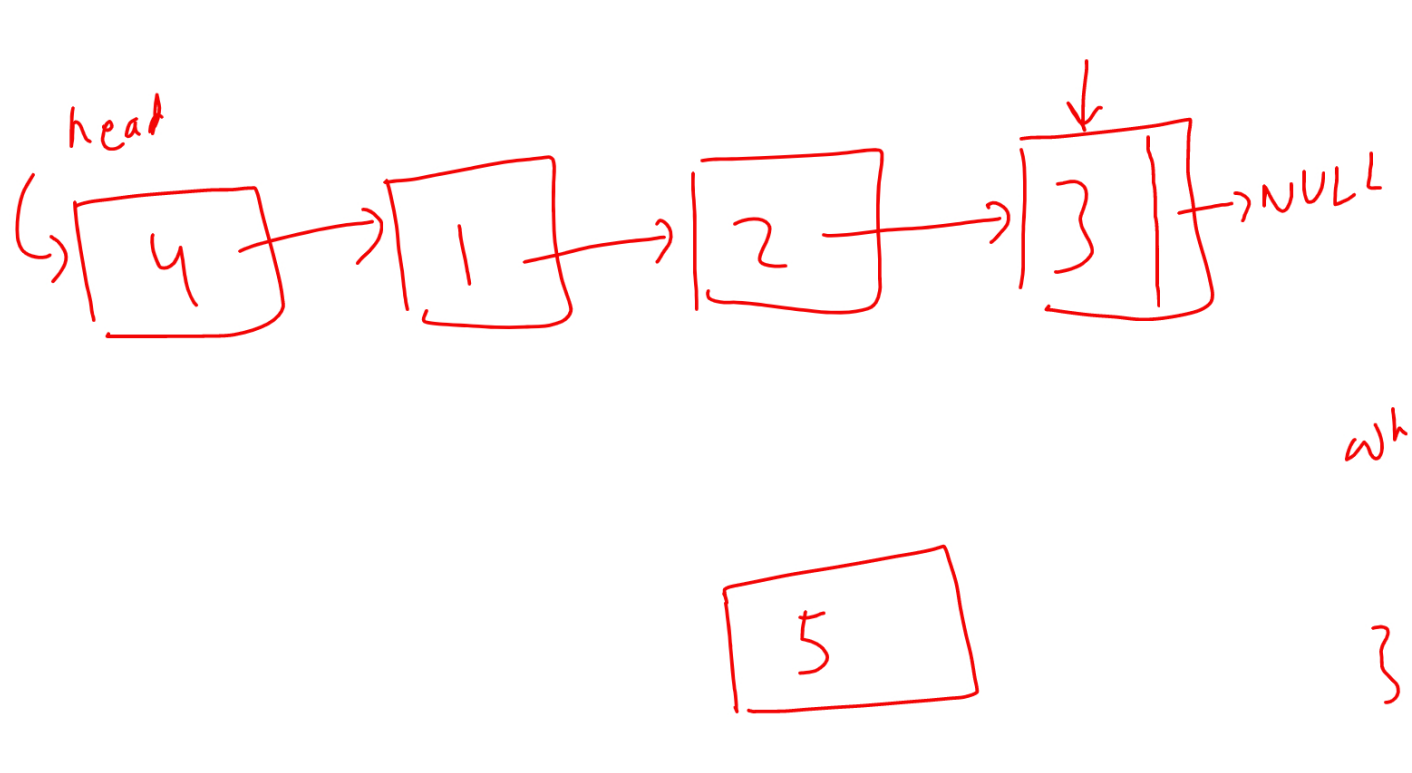
(cur→next != NULL) ✗

```
cur = head;
while (cur != NULL) {
  // ~~---~~ ------
   cur = cur→next;
}
```


1
2
3
NULL
~~head~~
4
new_node
head

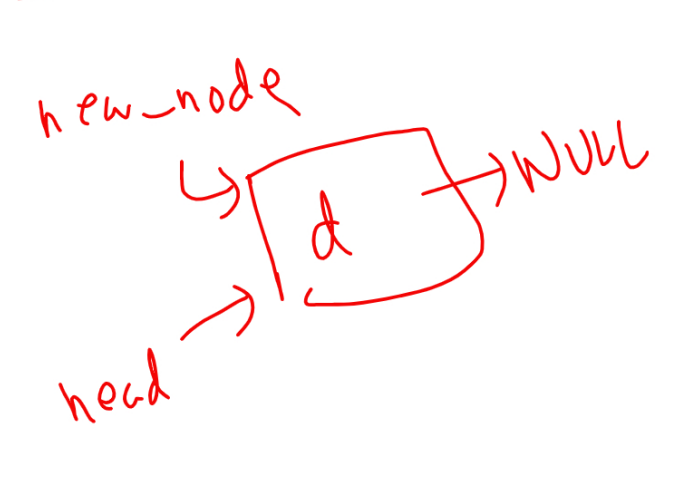
(d)

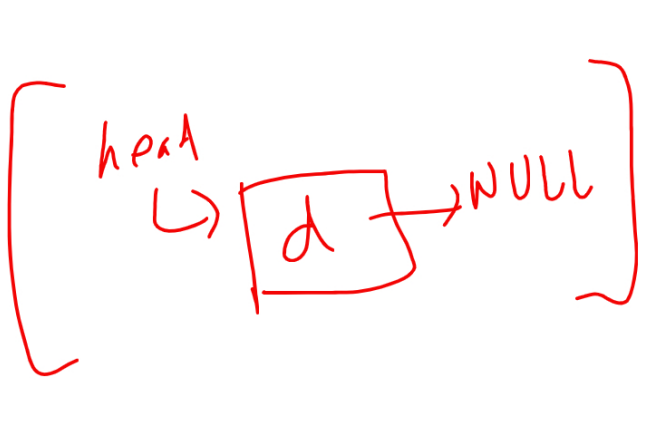
```
Node* new_node = new Node(d);
new_node→next = head;
head = new_node;
```

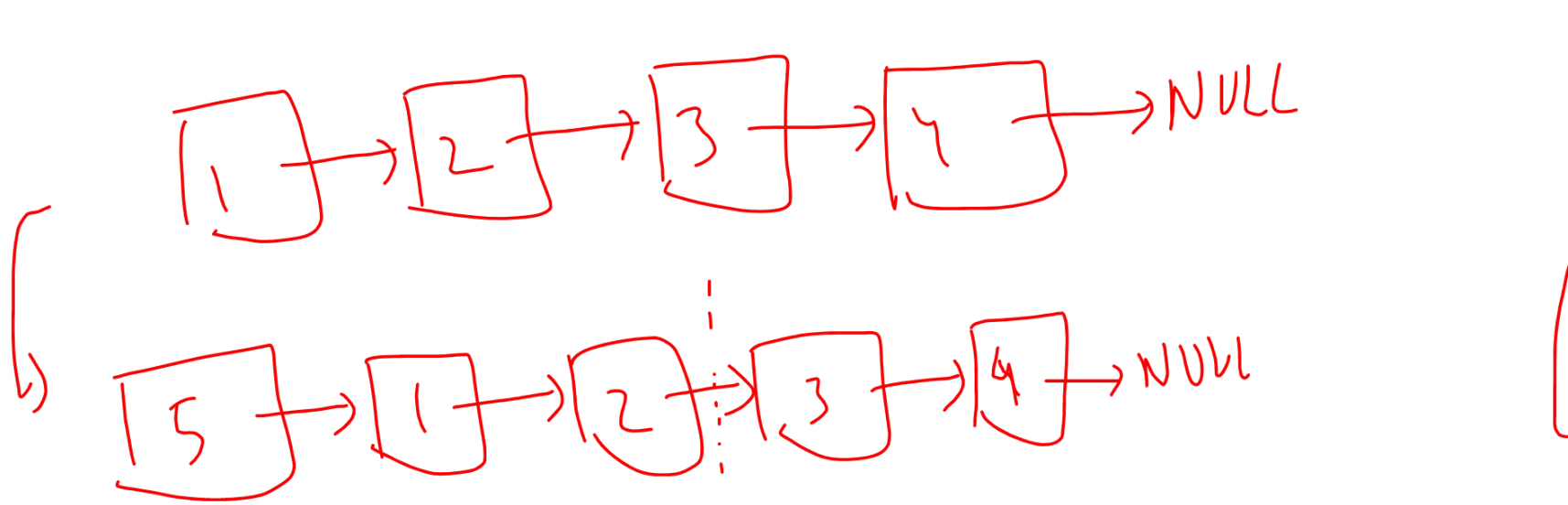

head
4
1
2
3
NULL
5

```
while (cur→next != NULL) {
   cur = cur→next;
}
cur→next = new Node(d);
```

head = NULL

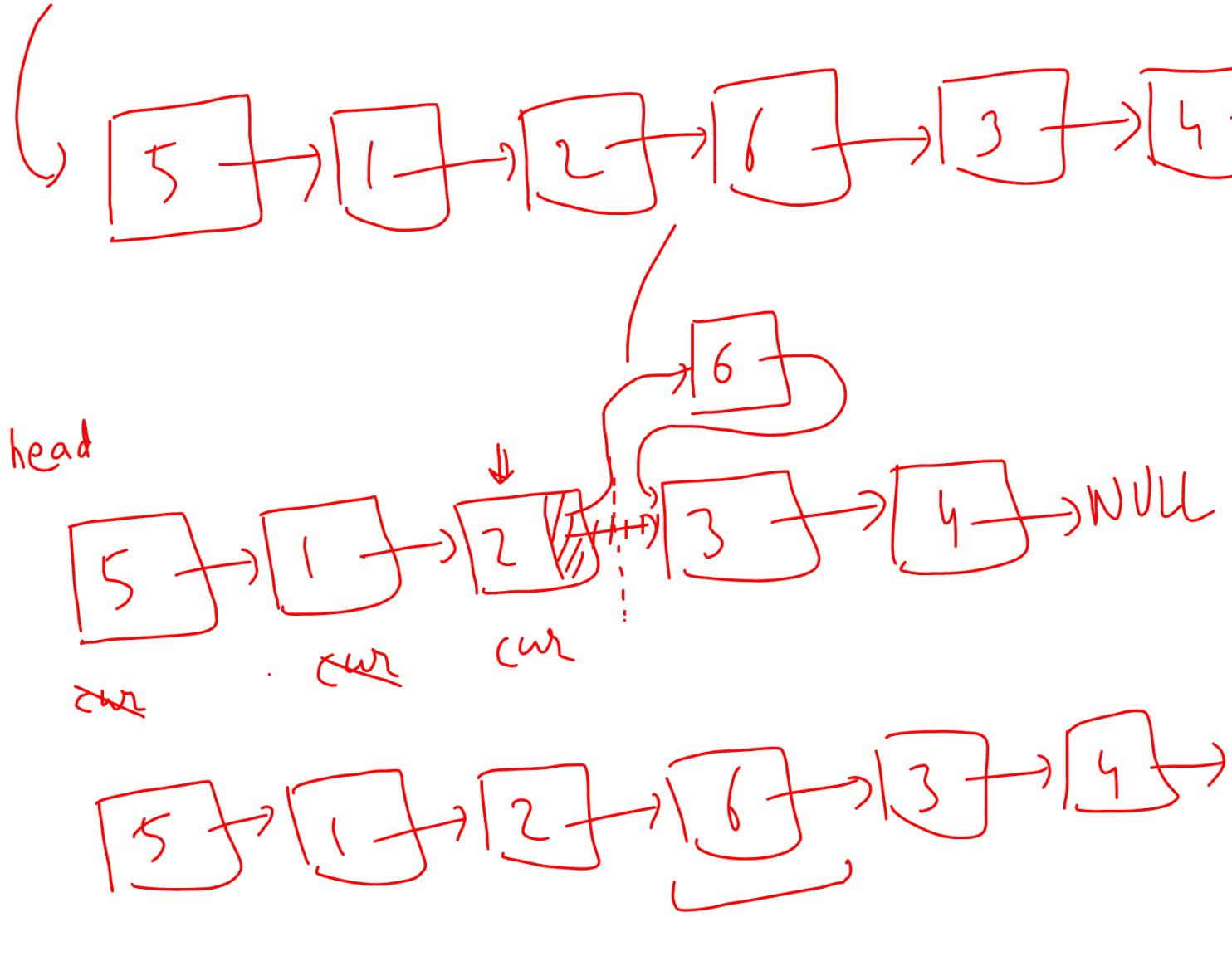

new_node
d
NULL
head



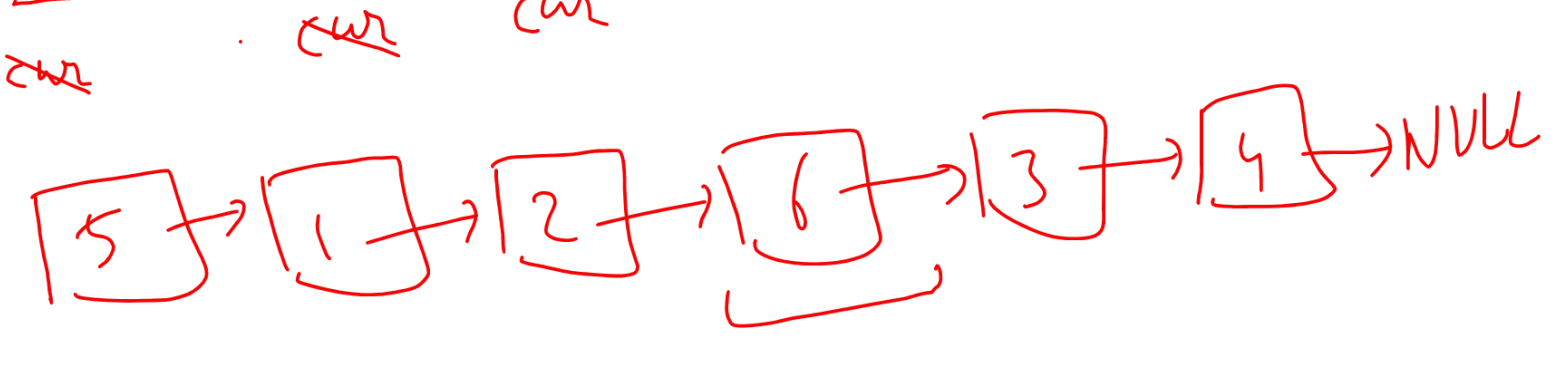

1
2
3
4
NULL
5
1
2
3
4
NULL
5
1
2
6
3
4
NULL

pos = 0, d = 5

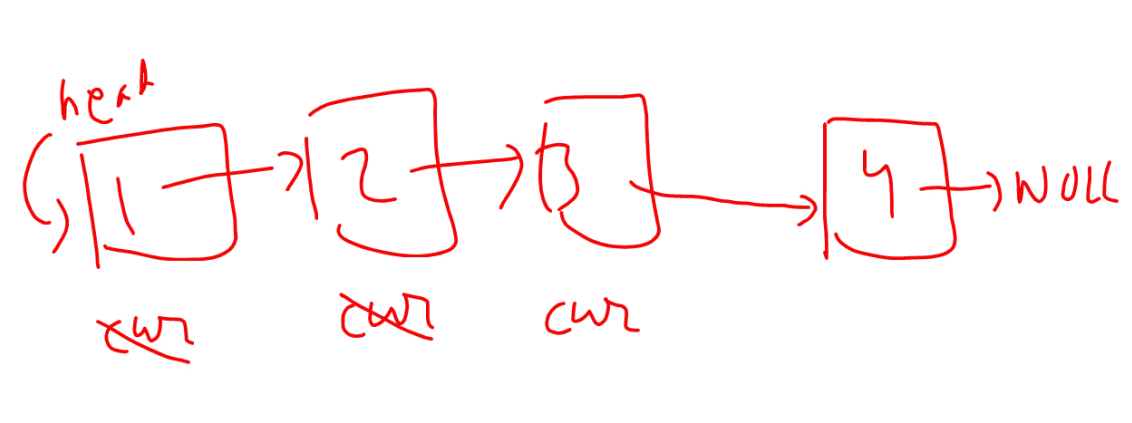
[pos = 3, d = 6]


6
head
5
1
2
3
4
NULL
~~cur~~
~~cur~~
cur

pos = 3, d = 6

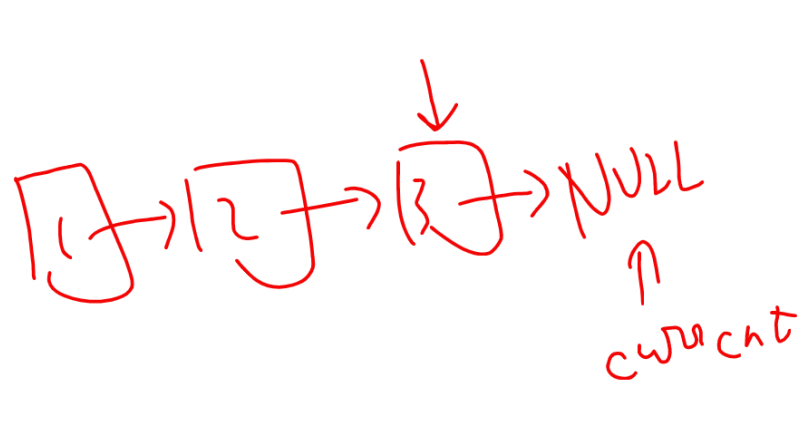

5
1
2
6
3
4
NULL

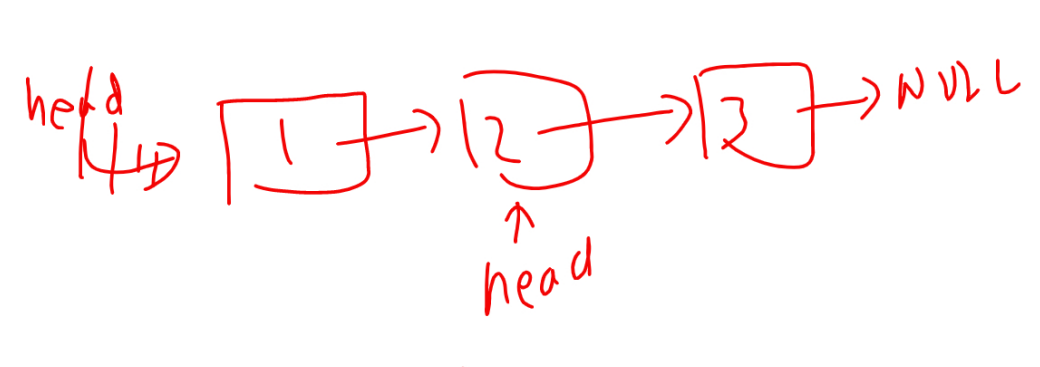
pos = ~~3~~ ~~2~~ 1


head
1
2
3
4
NULL
~~cur~~
~~cur~~
cur

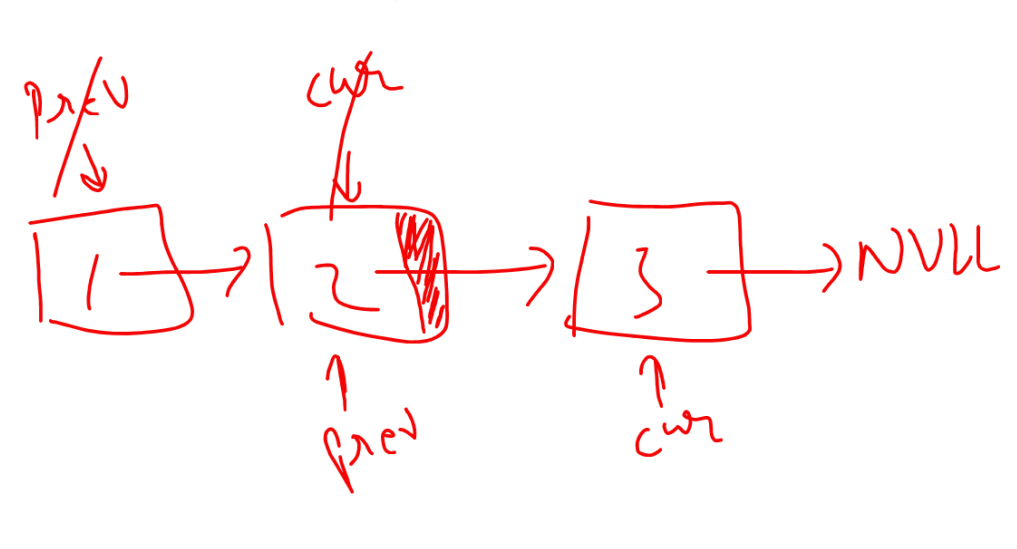
pos = ~~3~~ d = 4
~~2~~
1

pos = 4

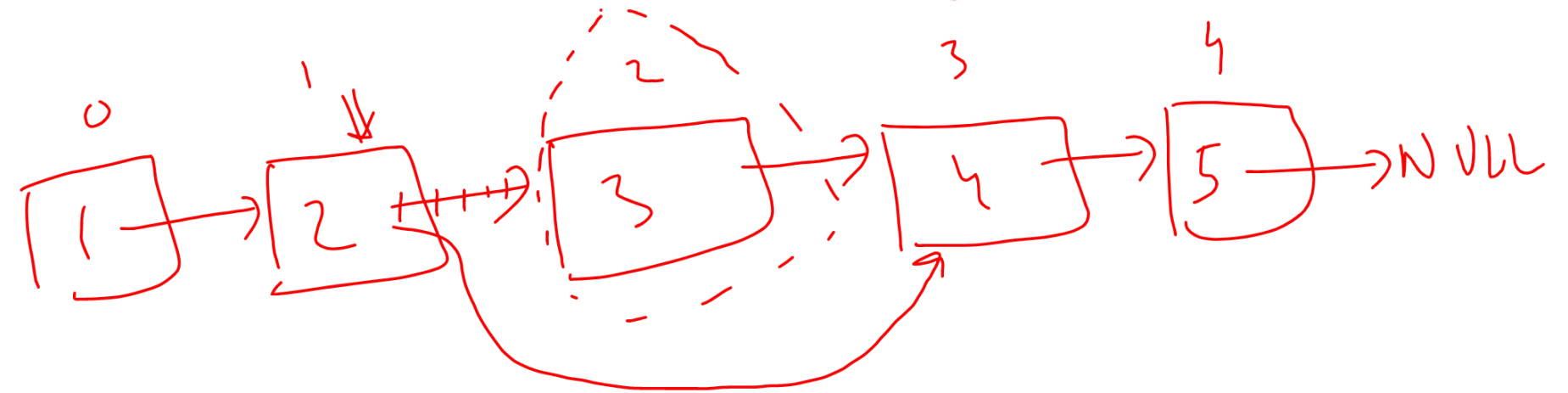

1
2
3
NULL
current


~~head~~
1
2
3
NULL
head

head = head→next.

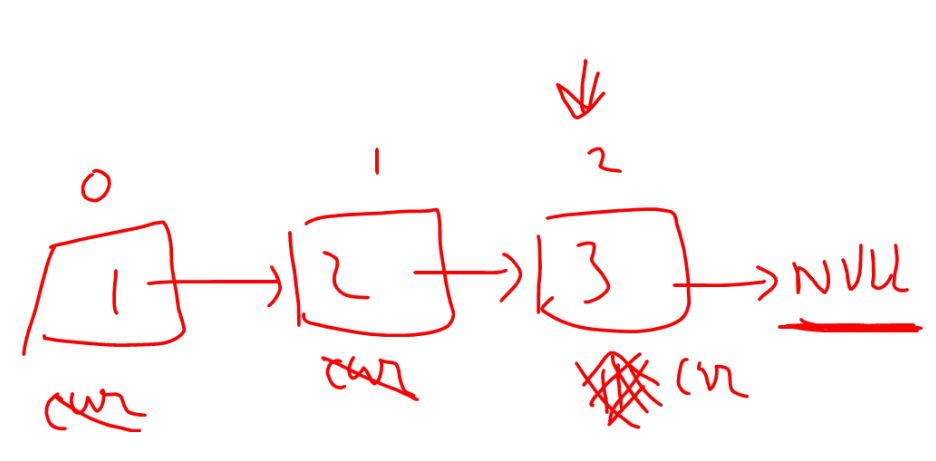

~~prev~~
cur
1
2
3
NULL
prev
cur

while (cur→next→next != NULL)


0
1
2
3
4
1
2
3
4
5
NULL

pos = 2

cur = cur→next


0
1
2
1
2
3
NULL
~~cur~~
~~cur~~
~~cur~~ cur

pos = ~~3~~ ~~2~~ 1

[pos > 2]
⇓
don't delete any node