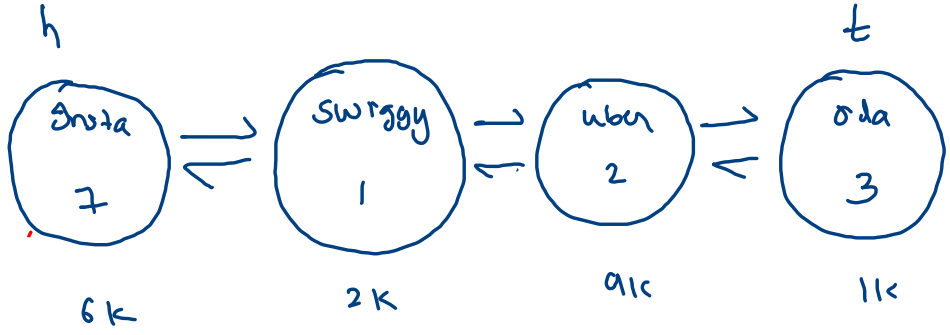
146. LRU Cache

limit : 4

hm string vs Node

ola : 11K
swiggy : 2K
insta : 6K
uber : 9K



put -> whatsapp, 5

put -> swiggy, 6

put -> insta, 7

put -> swiggy, 1

put -> uber, 2

put -> ola, 3

Node :

String appname;

int appstatus;

Node prev;

Node next;

limit : 4

put -> whatsapp, 5
put -> swiggy, 6
put -> insta, 7
put -> swiggy, 1
put -> uber, 2
put -> ola 3

```
if (app is not present) {
    if (cache.size == limit) -> rF, aL(new.app);
    else -> addLast (new_app)
}
else {
    remove(app);
    addLast(app);
}
```

```
["LRUCache", "put", "put", "get", "put", "get", "put", "get", "get", "get"]
[[2], [1, 1], [2, 2], [1], [3, 3], [2], [4, 4], [1], [3], [4]]
```

Output

```
[null, null, null, 1, null, -1, null, -1, 3, 4]
```

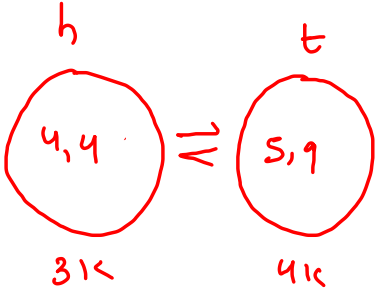
put [5,9]

get [3]

get [5]

null -1 9

limit = 2



hm

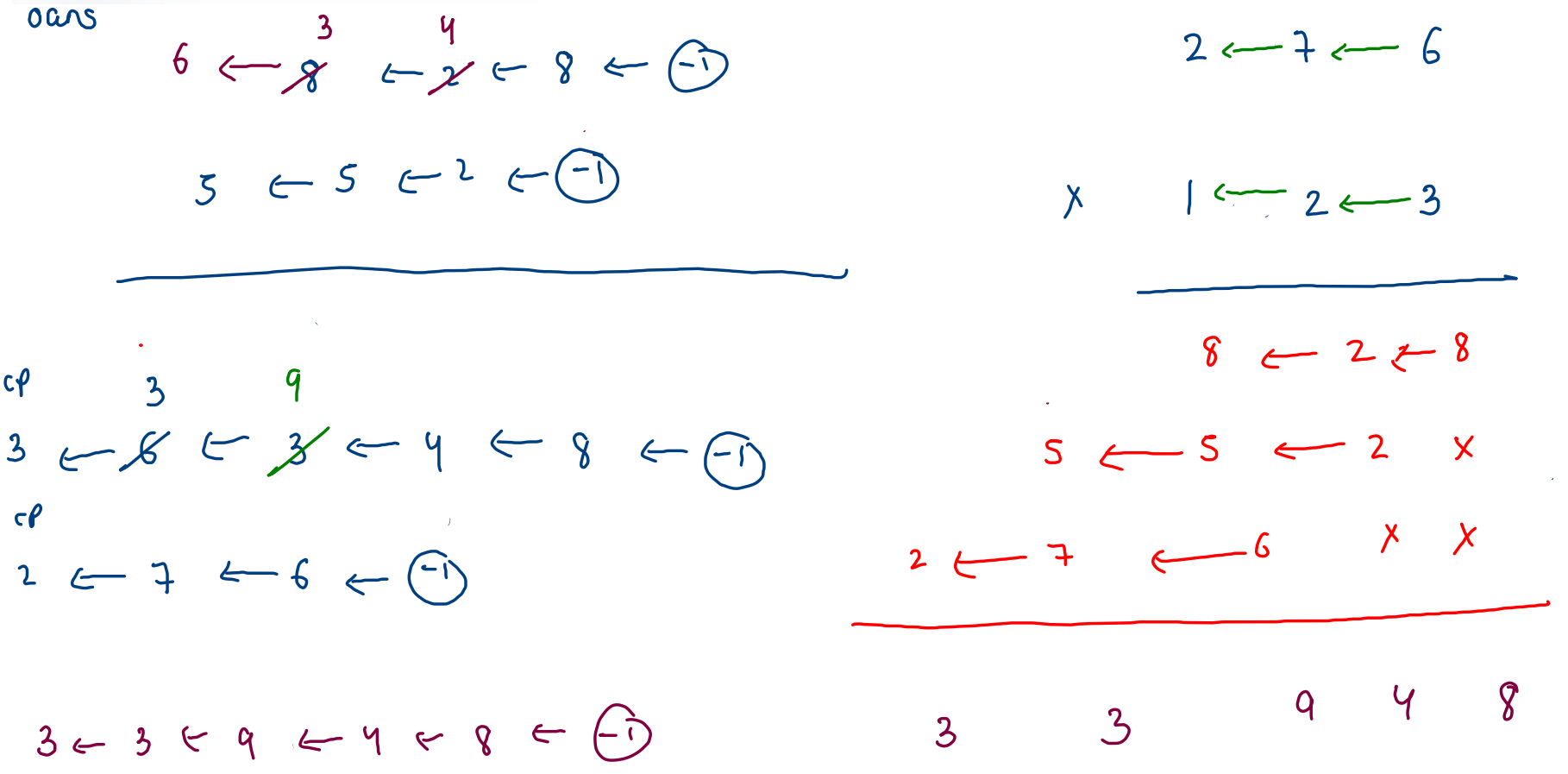
5 -> 4th

4 -> 3rd

```
public int get(int key) {
    //key -> app name
     if(map.containsKey(key) == false) {
        //app is not present yet
        return -1;
    }
    else {
        //app is already present
        ListNode node = map.get(key);

        //make this app most recent
        remove(node);
        addLast(node);

        return node.value;
    }
}
```

```
public void put(int key, int value) {
    //key -> app name
    //value -> app state

    if(map.containsKey(key) == false) {
        //app is not present yet

        ListNode node = new ListNode(key,value);
        map.put(key,node);

        addLast(node);

        if(size > limit) {
            //delete the least recent app
            int hd = removeFirst(); //head's data
            map.remove(hd);
        }
    }
    else {
        //app is already present
        ListNode node = map.get(key);

        //make this app most recent
        remove(node);
        addLast(node);

        node.value = value;
    }
}
```

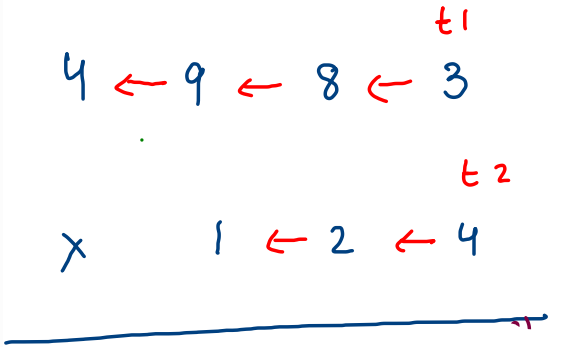
Multiply Two Linkedlist

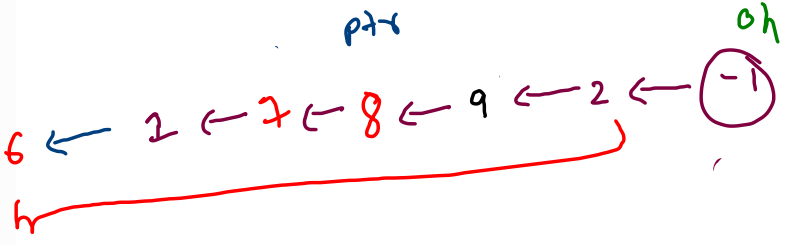


```java
public static void addition(ListNode ptr,ListNode ch) {
    //oh -> overall ans head
    //ch -> current linked list head

    //task  overall ans = overall ans + current ans

    ListNode op = ptr;
    ListNode cp = ch;

    int carry = 0;

    while(op.next != null || cp.next != null || carry != 0) {
        int sum = carry;

        if(op.next != null) {
            sum += op.next.val;
        }

        if(cp.next != null) {
            sum += cp.next.val;
            cp = cp.next;
        }

        int val = sum % 10;
        carry = sum / 10;

        if(op.next != null) {
            op.next.val = val;
            op = op.next;
        }
        else {
            op.next = new ListNode(val);
            op = op.next;
        }

    }
}
```

```java
public static ListNode multiplyTwoLL(ListNode l1, ListNode l2) {

    ListNode oh = new ListNode(-1);
    ListNode ot = oh;

    ListNode ptr = oh;

    ListNode t1 = reverse(l1);
    ListNode t2 = reverse(l2);

    while(t2 != null) {
        int d = t2.val;
        t2 = t2.next;

        ListNode sans = singleDigitMult(t1,d);
        addition(ptr,sans);

        ptr = ptr.next;
    }

    ListNode ans = reverse(oh.next);
    return ans;
}
```

```java
public static ListNode singleDigitMult(ListNode t1,int d) {
    ListNode dh = new ListNode(-1);
    ListNode dt = dh;

    int carry = 0;

    while(t1 != null || carry != 0) {
        int mult = carry;

        if(t1 != null) {
            mult += t1.val * d;
            t1 = t1.next;
        }

        int val = mult % 10;
        carry = mult / 10;

        ListNode nn = new ListNode(val);
        dt.next = nn;
        dt = dt.next;
    }

    return dh;
}
```





6 -> 1 -> 7 -> 8 -> 9 -> 2