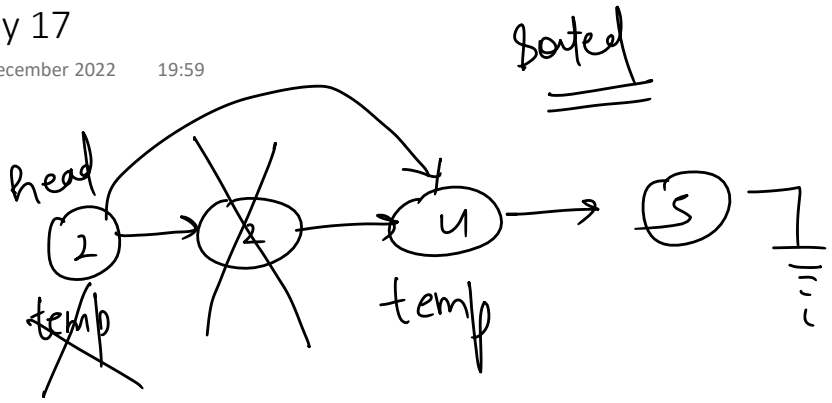
Day 17

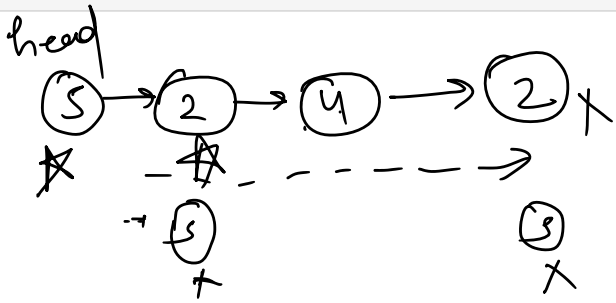
04 December 2022 19:59

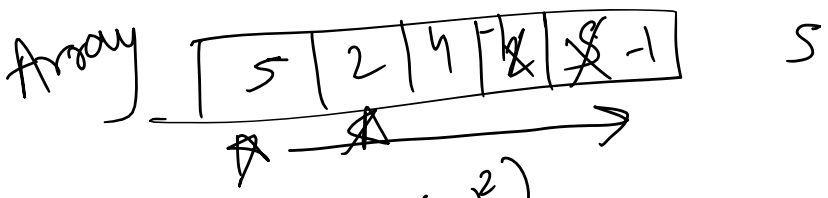


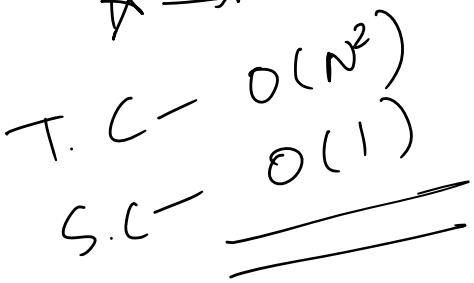
Given an unsorted linked list of **N** nodes. The task is to remove duplicate elements from this unsorted Linked List. When a value appears in multiple nodes, the node which appeared first should be kept, all others duplicates are to be removed.

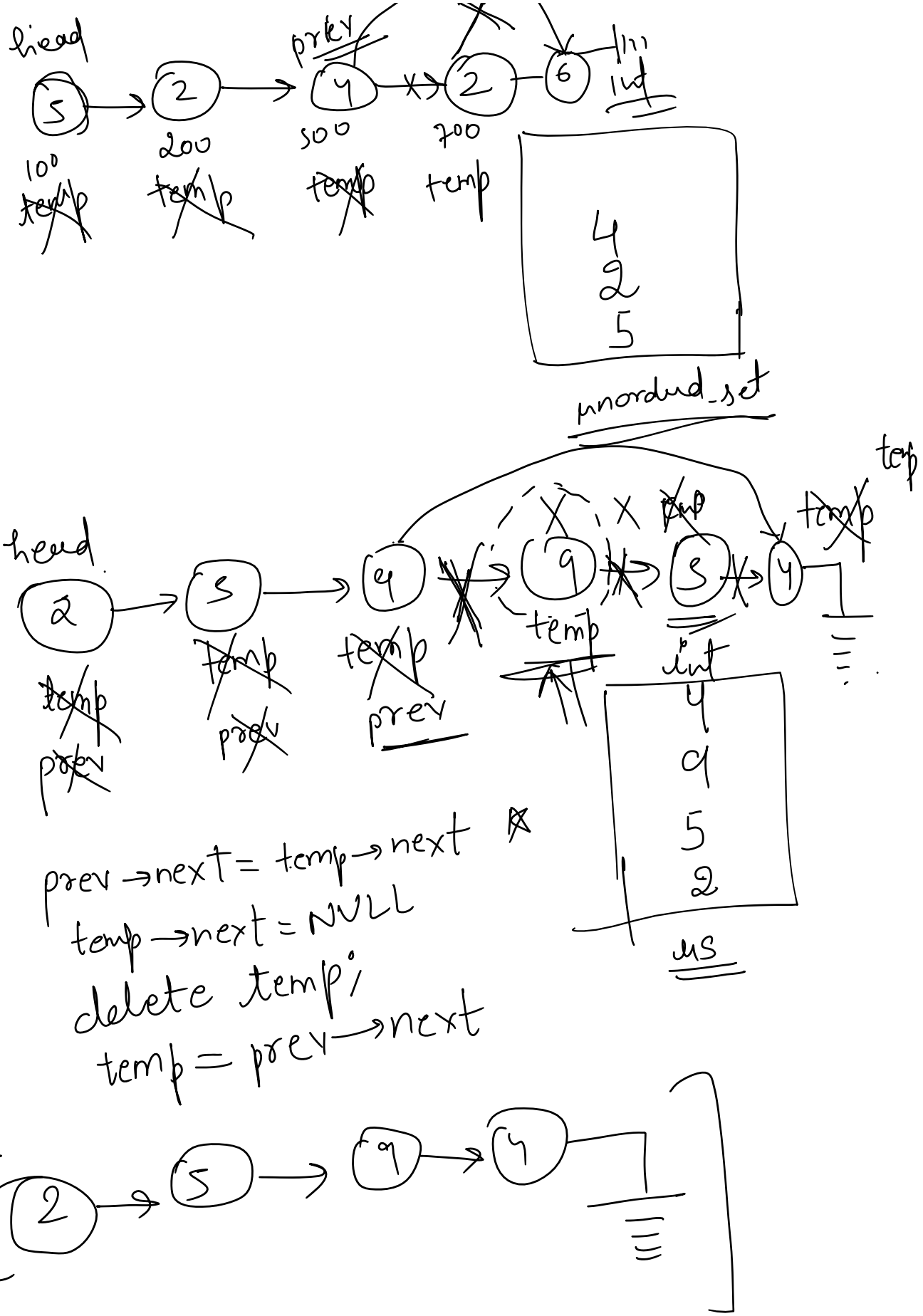
Example 1:

Input:
N = 4
value[] = {5,2,2,4}
Output: 5 2 4
Explanation: Given linked list elements are 5->2->2->4, in which 2 is repeated only. So, we will delete the extra repeated elements 2 from the linked list and the resultant linked list will contain 5->2->4

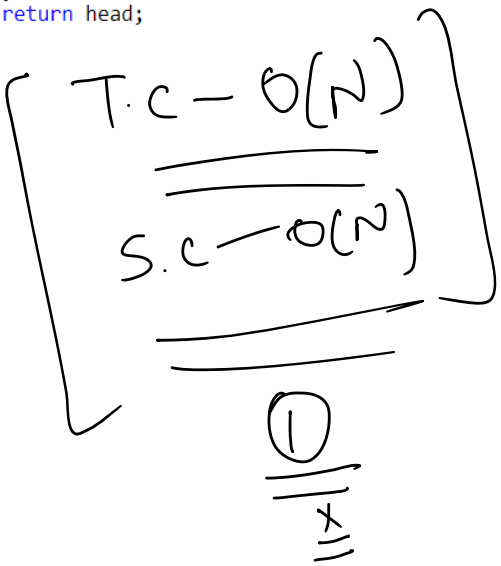




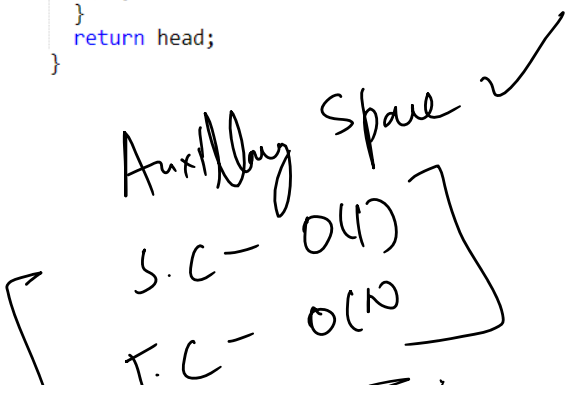



head
prev
temp
100
200
500
700
int
4
2
5
unordered_set
head
prev
temp
int
4
9
5
2
us
prev->next = temp->next
temp->next = NULL
delete temp;
temp = prev->next

```
public:
//Function to remove duplicates from unsorted linked list.
Node * removeDuplicates( Node *head)
{
  unordered_set<int> checker;
  Node *prev = NULL, *temp = head;
  while(temp!= NULL){
      if(checker.find(temp->data)!=checker.end()){
          prev->next = temp->next;
          temp->next = NULL;
          delete temp;
          temp = prev->next;
      }else{
          checker.insert(temp->data);
          prev = temp;
          temp = temp->next;
      }
  }
  return head;
}
```




```
//Function to remove duplicates from unsorted link
Node * removeDuplicates( Node *head)
{
  vector<int> hashmap(10001,0);
  Node *prev = NULL, *temp = head;
  while(temp!= NULL){
      if(hashmap[temp->data] == 1){
          prev->next = temp->next;
          temp->next = NULL;
          delete temp;
          temp = prev->next;
      }else{
          hashmap[temp->data] = 1;
          prev = temp;
          temp = temp->next;
      }
  }
  return head;
}
```

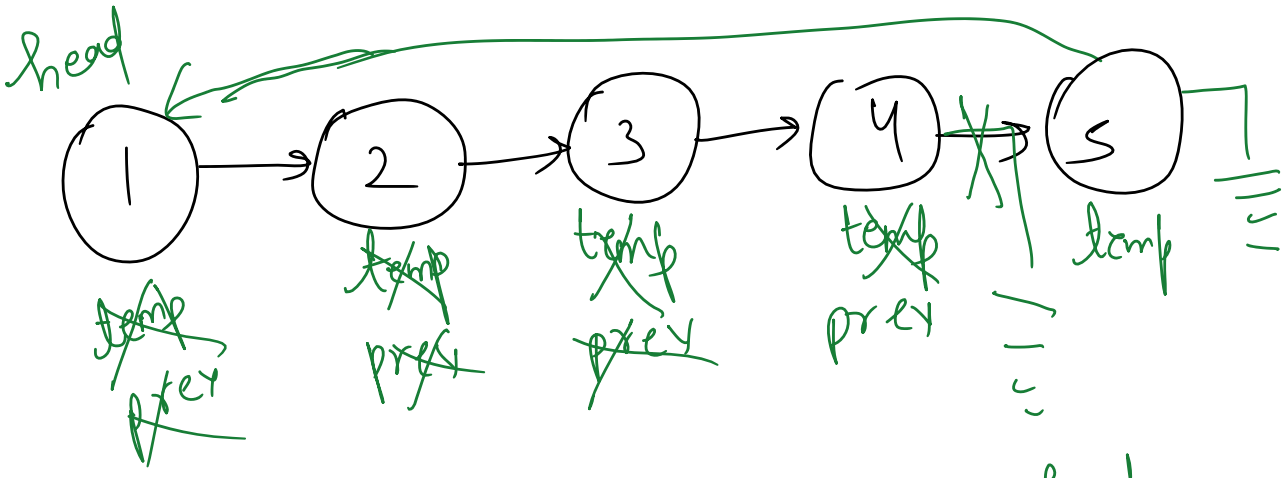


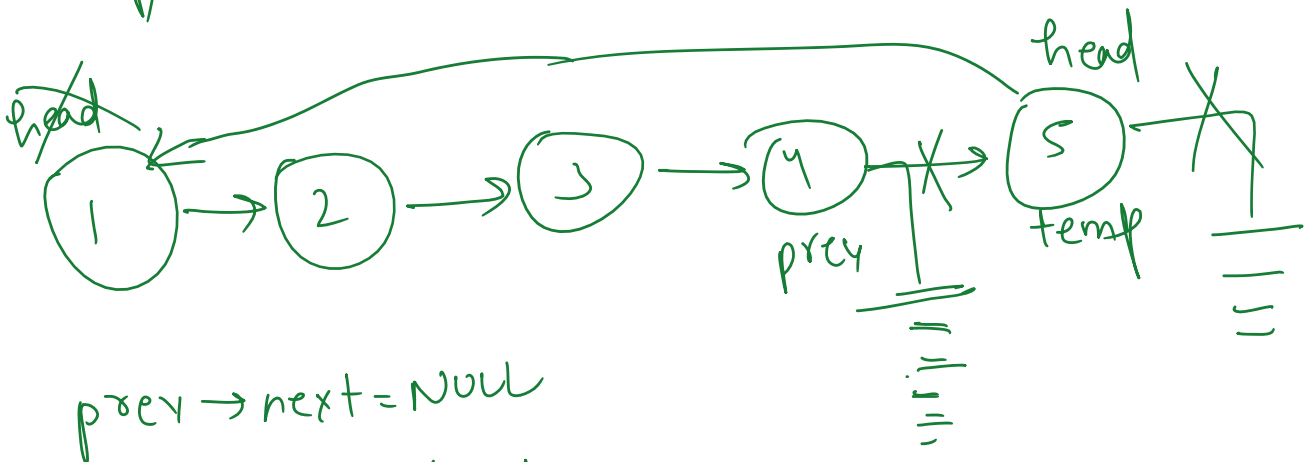
T.C - O(N)

Move last element to front

I/P 1 → 2 → 3 → 4 → 5

O/P 5 → 1 → 2 → 3 → 4 //

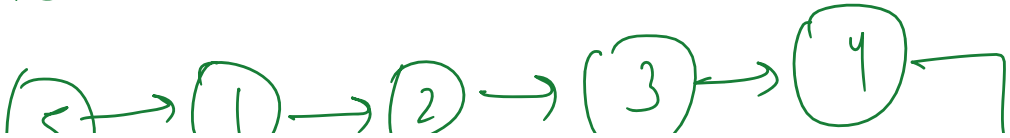


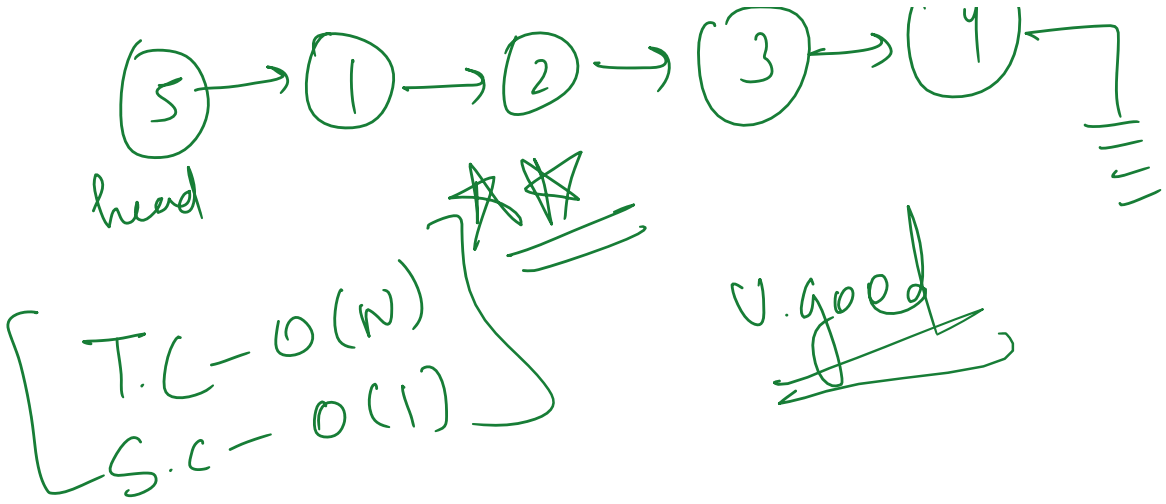


prev → next = NULL

temp → next = head

head = temp





Add 1 to a number represented as linked list

Easy Accuracy: **31.91%** Submissions: **100k+** Points: **2**

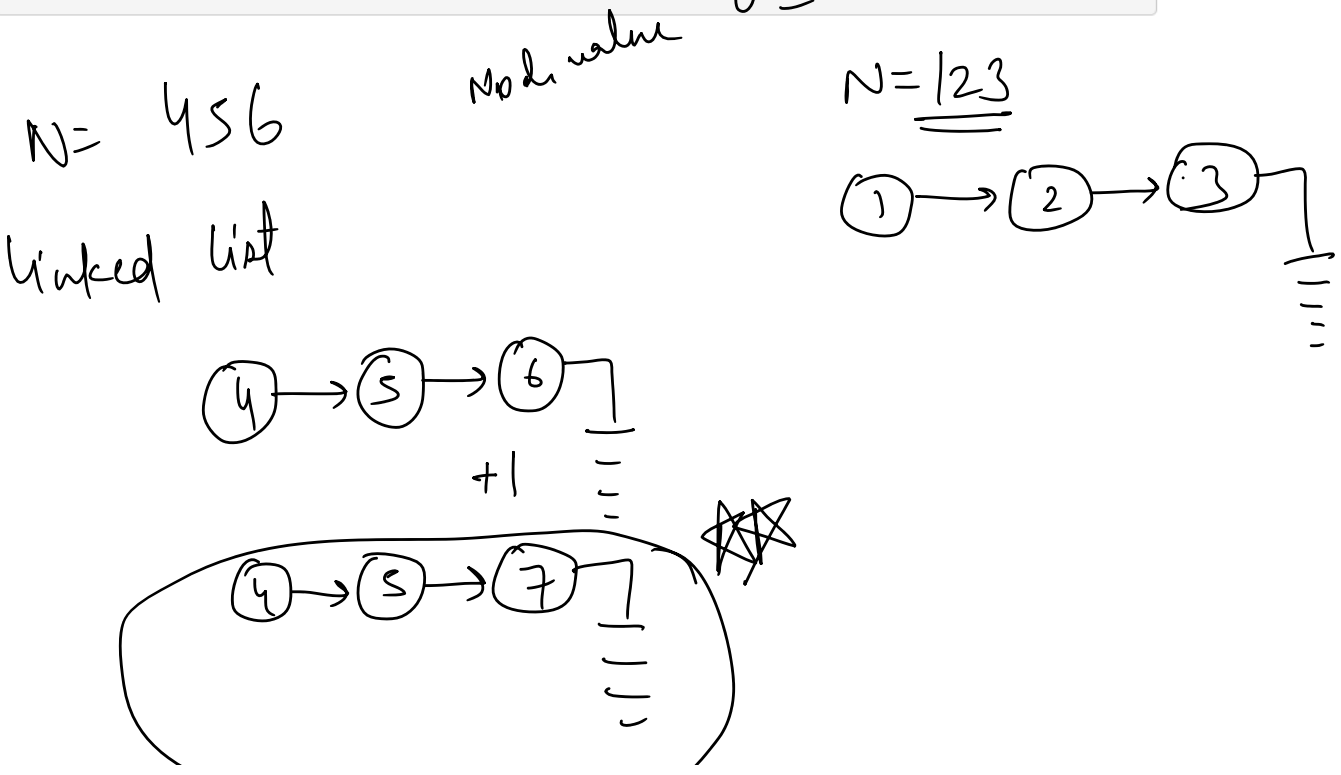
A number **N** is represented in Linked List such that each digit corresponds to a node in linked list. You need to add 1 to it.

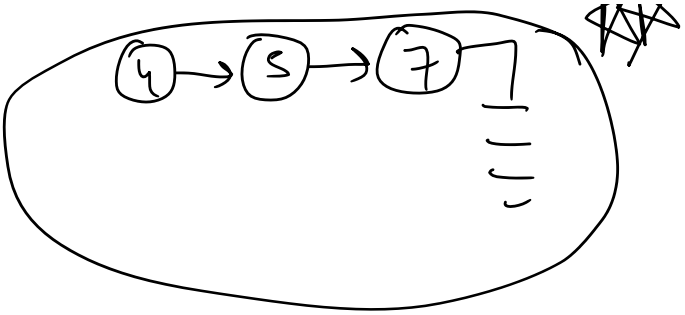
Example 1:

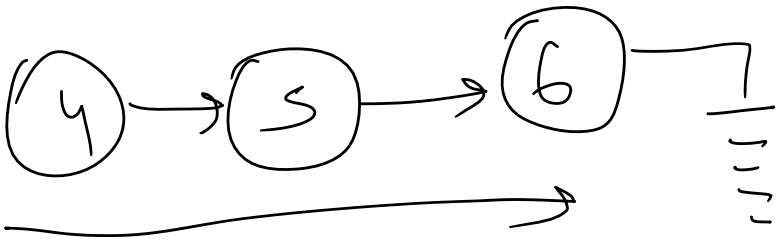
Input:
LinkedList: 4->5->6
Output: 457

Example 2:

Input:
LinkedList: 1->2->3
Output: 124







3

$4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 = 456 + 1 = 457$

$n_1 = 457$

while ($n_1 != 0$)

$r = n_1 \% 10$

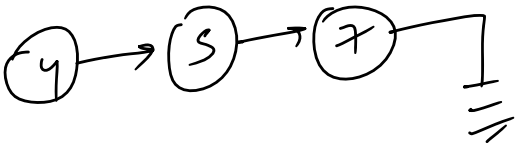
$rev = n_1 / 10$

n_1

y

$n_1 = *0$

reverse



Answer

Algorithm

(1) Count no. of digits $c = 456$ (3)

$\rightarrow 4 \times 10^{c-1} + 5 \times 10^{c-2} + 6 \times 10^{c-3}$

$4\times10^2 + 5\times10^1 + 6\times10^0 = 756$

$n_1 = 457$

⇓ linked list representation

999 + 1

45 % 10 = 5 4

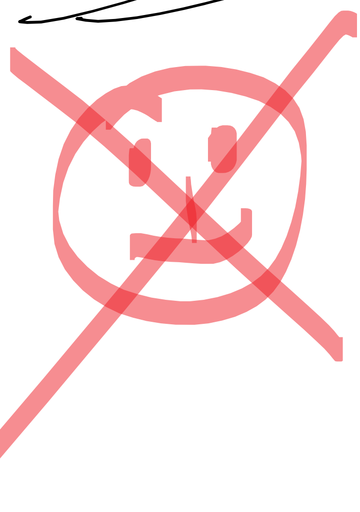
% 10

Linked list + 2 numerical

7 → 5 → 4

⇓ reverse

457



★ Linked list changes!!

2 → 1 →

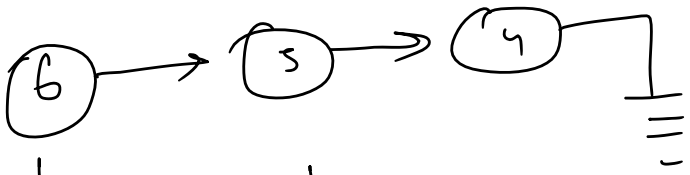
-2^{31} to 2^{31} ★

Algorithm

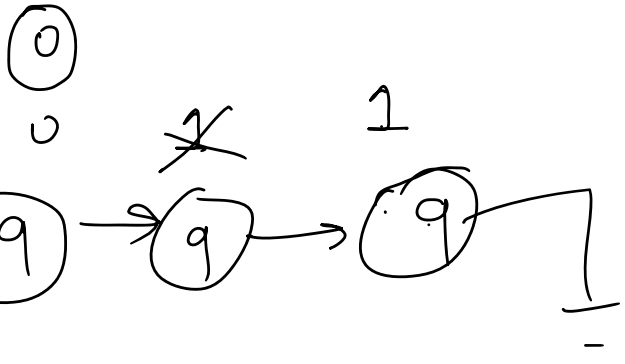
456
+ 1
457

456
+ 7
463

(1) reverse Linked Linked



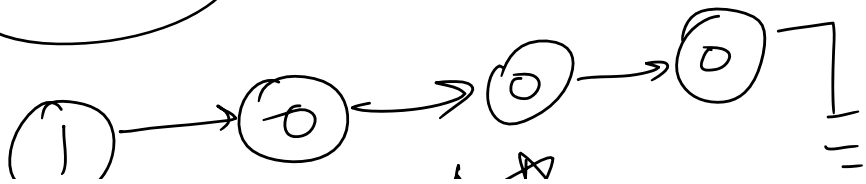
+1 +1

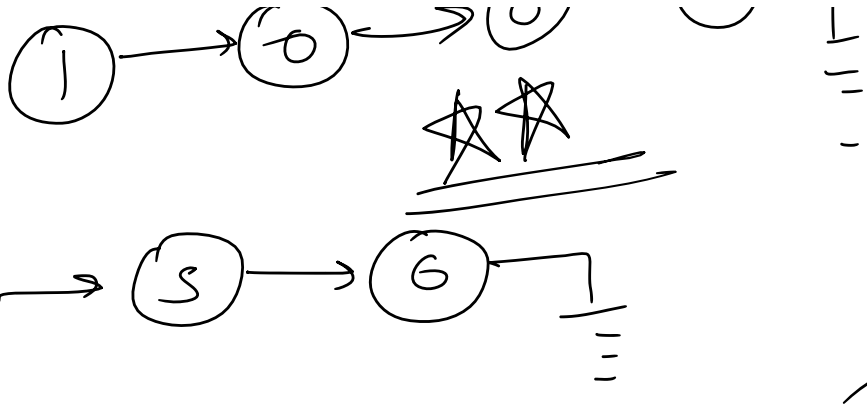


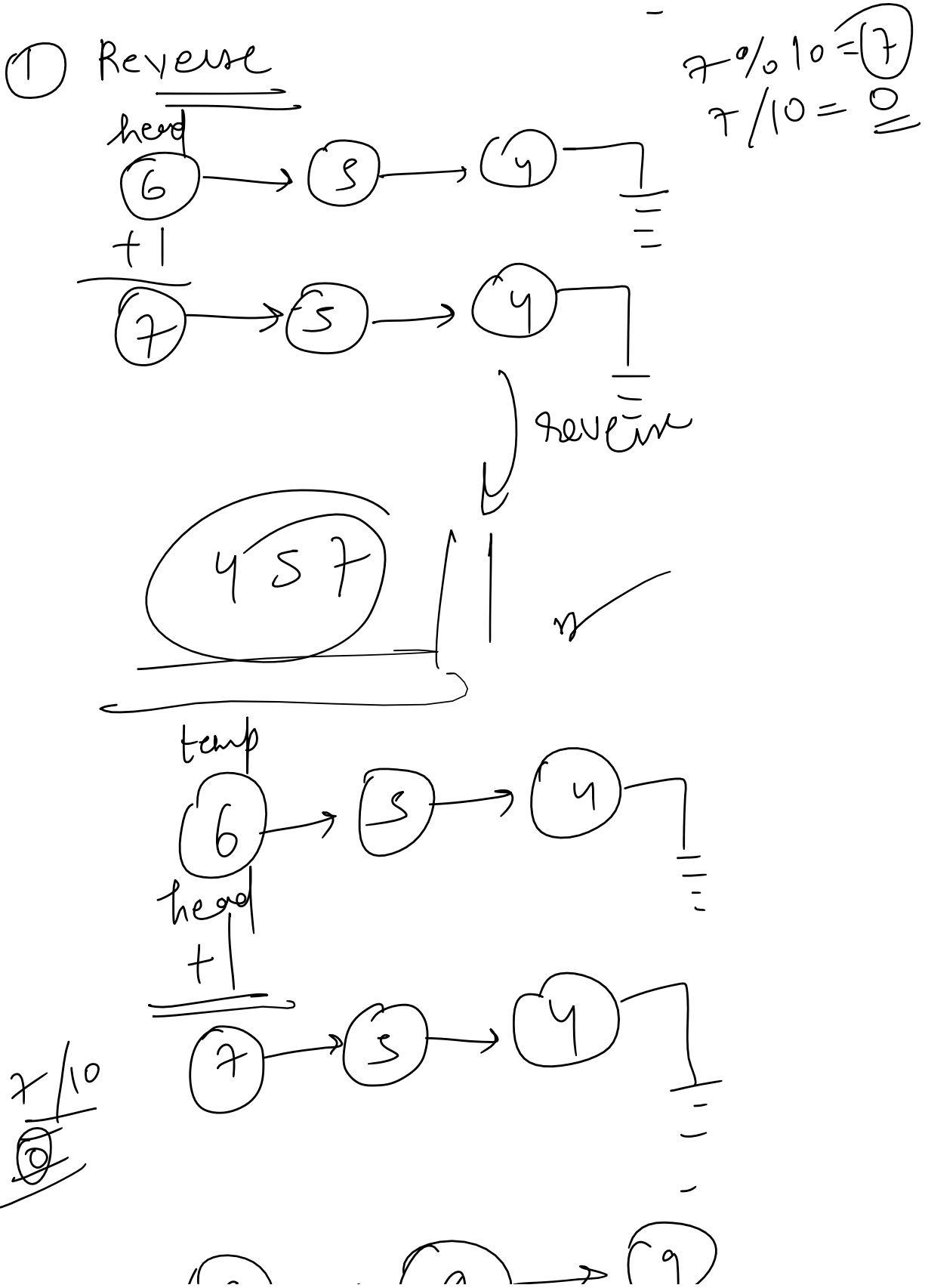
10 % 10 = 0
10/10 = 1
carry = 1

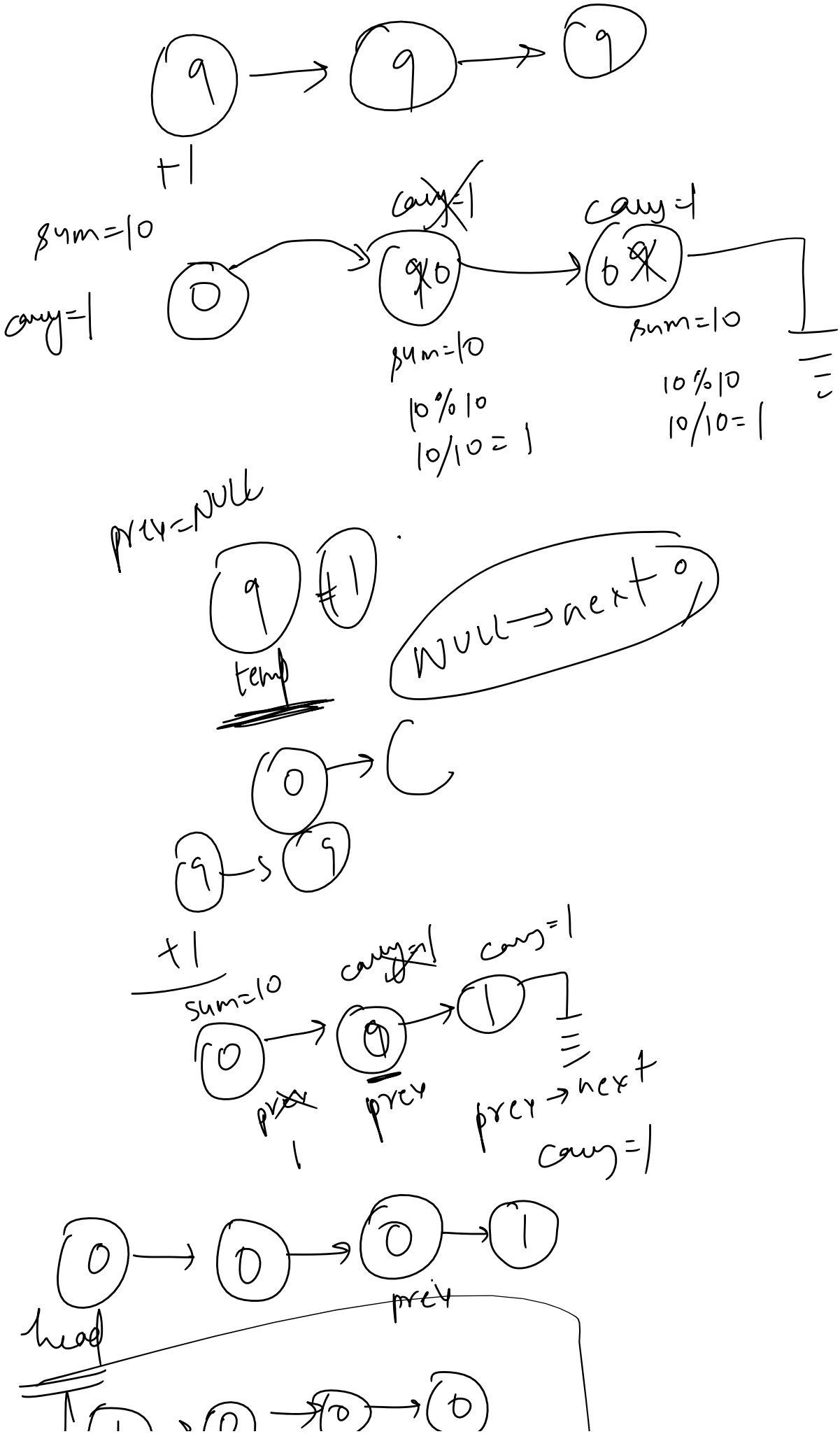
0001

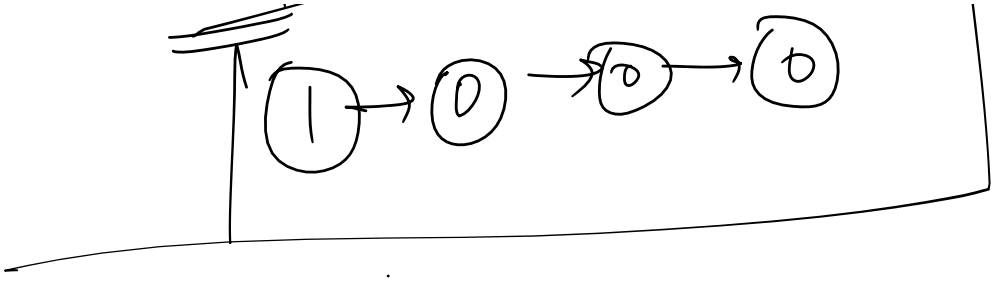
reverse





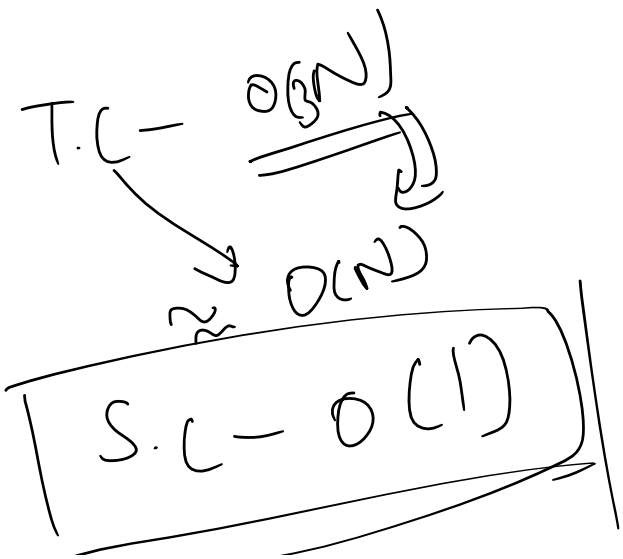

1) Reverse
head
+1
reverse
temp
head
+1
x/10

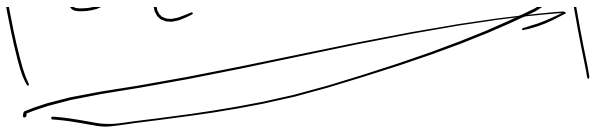

+1
sum=10
carry=1
carry=1
sum=10
10%10
10/10=1
sum=10
10%10
10/10=1
prev=NULL
temp
NULL→next=0
+1
sum=10
carry=1
carry=1
prev
prev
prev→next
carry=1
head
prev



```
//{ Driver Code Starts
//Initial template for C++

#include <bits/stdc++.h>
using namespace std;

struct Node
{
   int data;
   struct Node* next;

   Node(int x){
      data = x;
      next = NULL;
   }
};

void printList(Node* node)
{
   while (node != NULL) {
      cout << node->data;
      node = node->next;
   }
   cout<<"\n";
}


// } Driver Code Ends
//User function template for C++

/*

struct Node
{
   int data;
   struct Node* next;

   Node(int x){
      data = x;
      next = NULL;
   }
};

*/

class Solution
{
   public:
   Node * reverse(Node *head){
      Node *prev = NULL,*curr = head,*next = NULL;
      while(curr != NULL){
         next = curr->next;
         curr->next = prev;
         prev = curr;
         curr = next;
      }
      return prev;
   }
   Node* addOne(Node *head)
   {
```

```
        Node*head2 = reverse(head);
        int carry  = 0;
        Node * temp = head2;
        Node *prev = NULL;
        while(temp != NULL){
            int sum = temp->data + carry;
            if(temp == head2){
                sum +=1;
            }
            temp->data = sum%10;
            carry = sum/10;
            if(carry == 0){
                break;
            }
            prev = temp;
            temp = temp->next;
        }
        if(carry != 0){
            if(prev == NULL){
                head2->next  = new Node(carry);
            }else{
                prev-> next = new Node(carry);
            }
        }
        return reverse(head2);
    }
};

//{ Driver Code Starts.

int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        string s;
        cin>>s;

        Node* head = new Node( s[0]-'0' );
        Node* tail = head;
        for(int i=1; i<s.size(); i++)
        {
            tail->next = new Node( s[i]-'0' );
            tail = tail->next;
        }
        Solution ob;
        head = ob.addOne(head);
        printList(head);
    }
    return 0;
}

// } Driver Code Ends
```



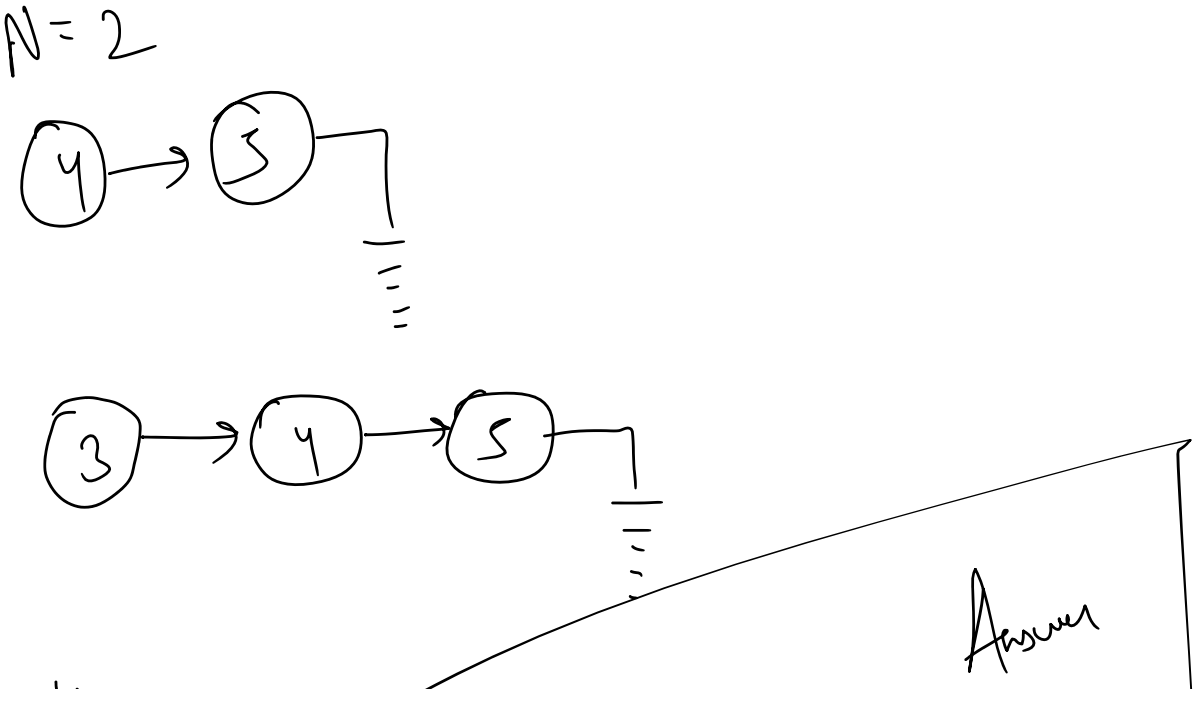


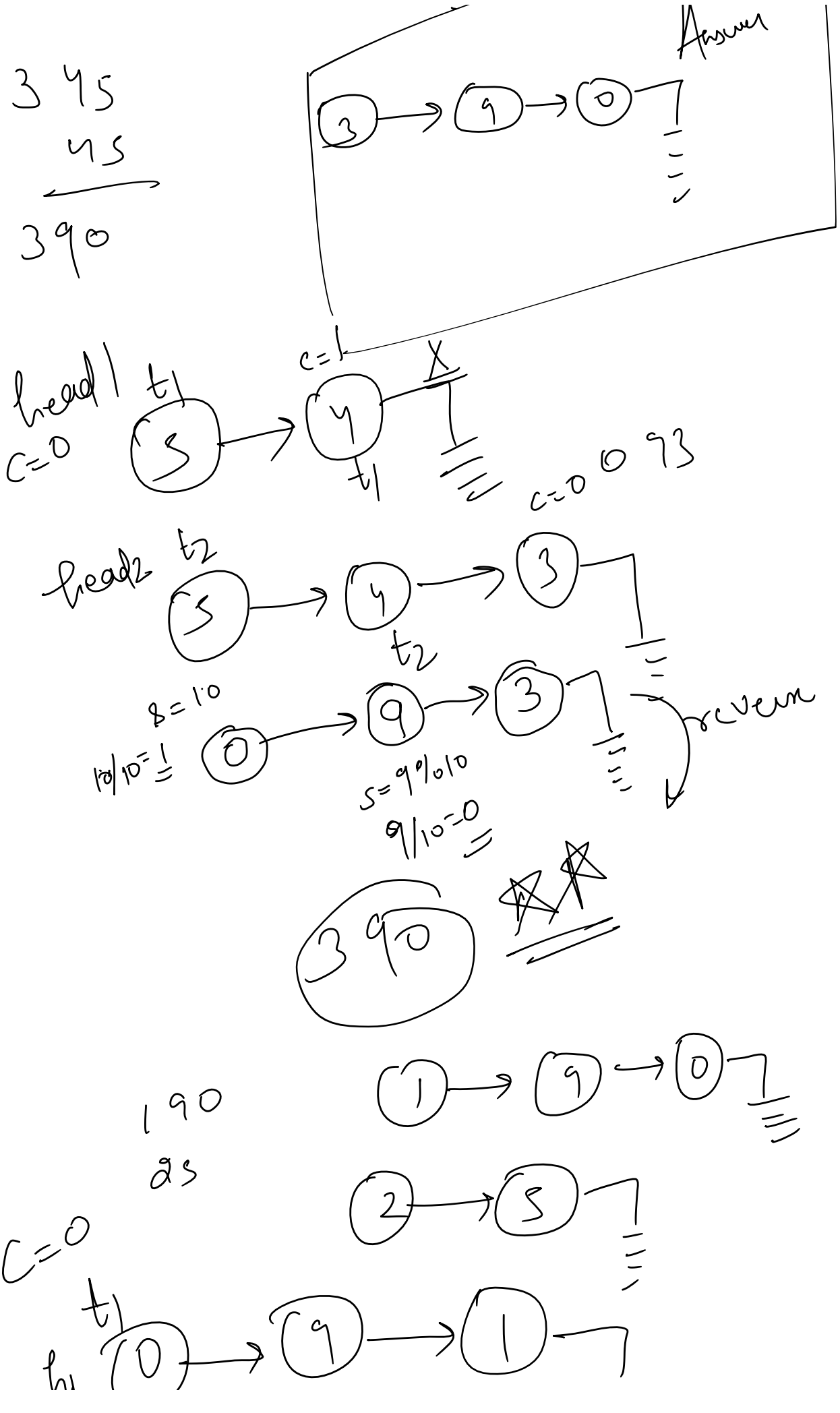
Given two decimal numbers represented by two linked lists of size **N** and **M** respectively. The task is to return a linked list that represents the sum of these two numbers.

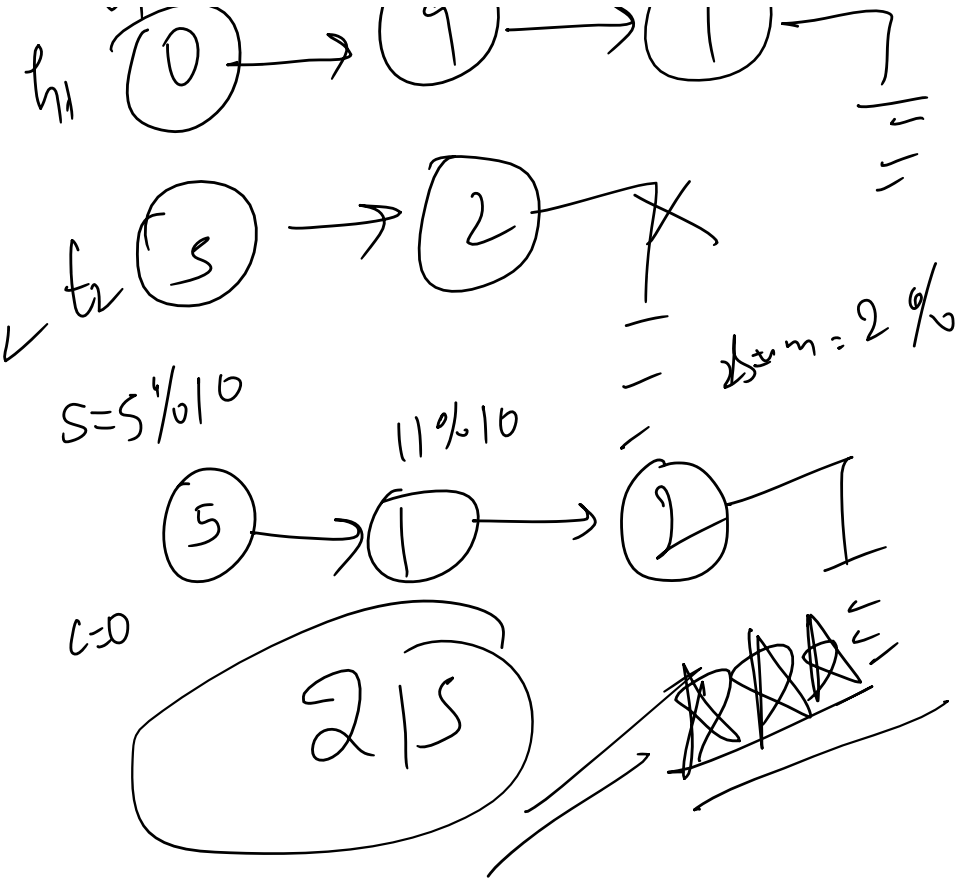
For example, the number **190** will be represented by the linked list, **1->9->0->null,** similarly **25** by **2->5->null.** Sum of these two numbers is 190 + 25 = **215,** which will be represented by **2->1->5->null.** You are required to return the head of the linked list **2->1->5->null.**

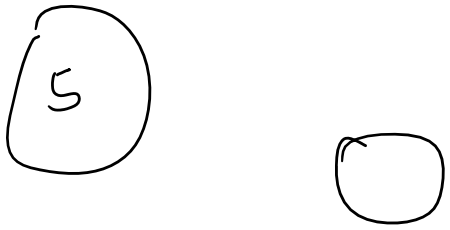
Example 1:

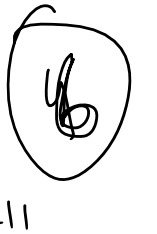
Input:
N = 2
valueN[] = {4,5}
M = 3
valueM[] = {3,4,5}
Output: 3 9 0
Explanation: For the given two linked list (4 5) and (3 4 5), after adding the two linked list resultant linked list will be (3 9 0).




Answer
3 4 5
4 5
3 9 0
head1 t1
c=0
c=1
X
t1
c=0 0 93
head2 t2
t2
s=10
10/10=1
s=9%10
9/10=0
return
3 9 0
1 9 0
as
c=0
t1
h1

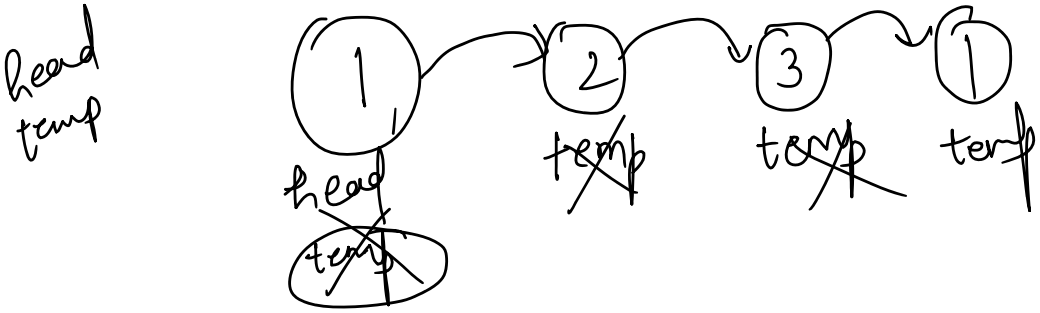

h1
0
7
1
h2
5
2
sum = 2%10
5=5%10
11%10
5
1
2
c=0
215

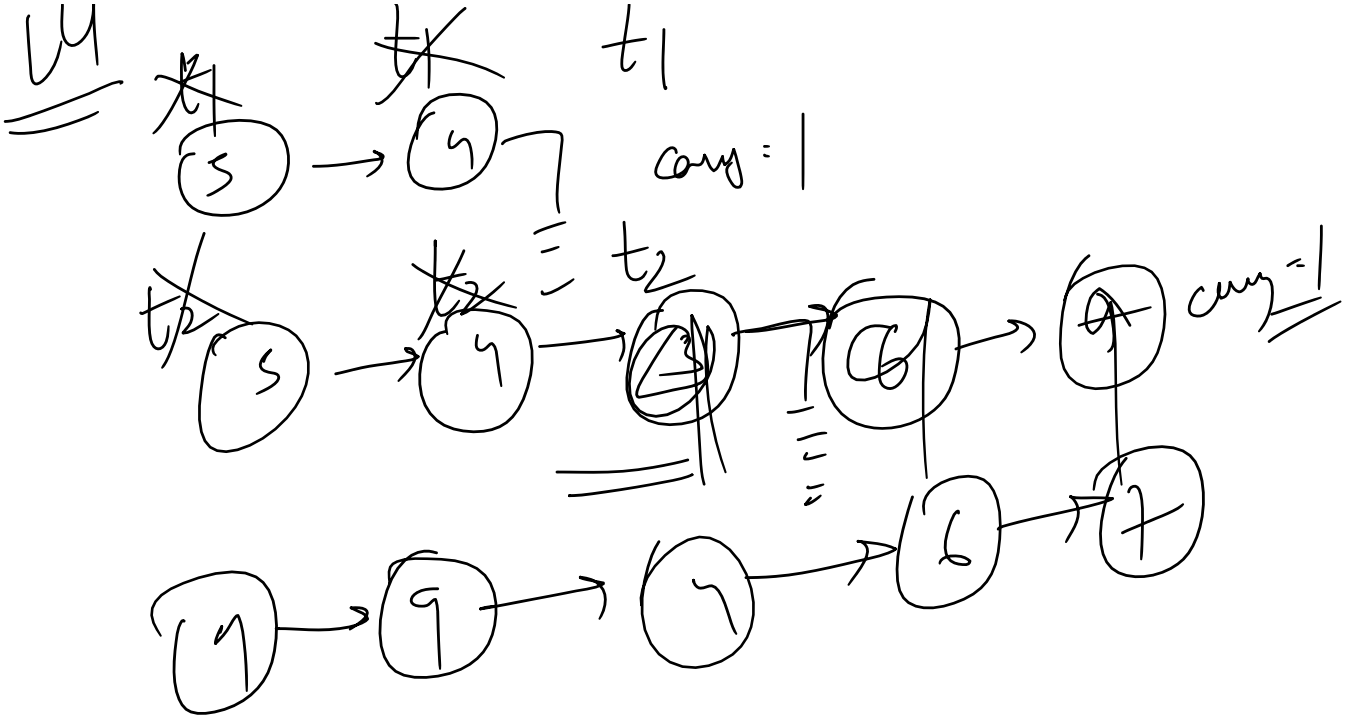

5


6

9 = 11

11%10 = 1

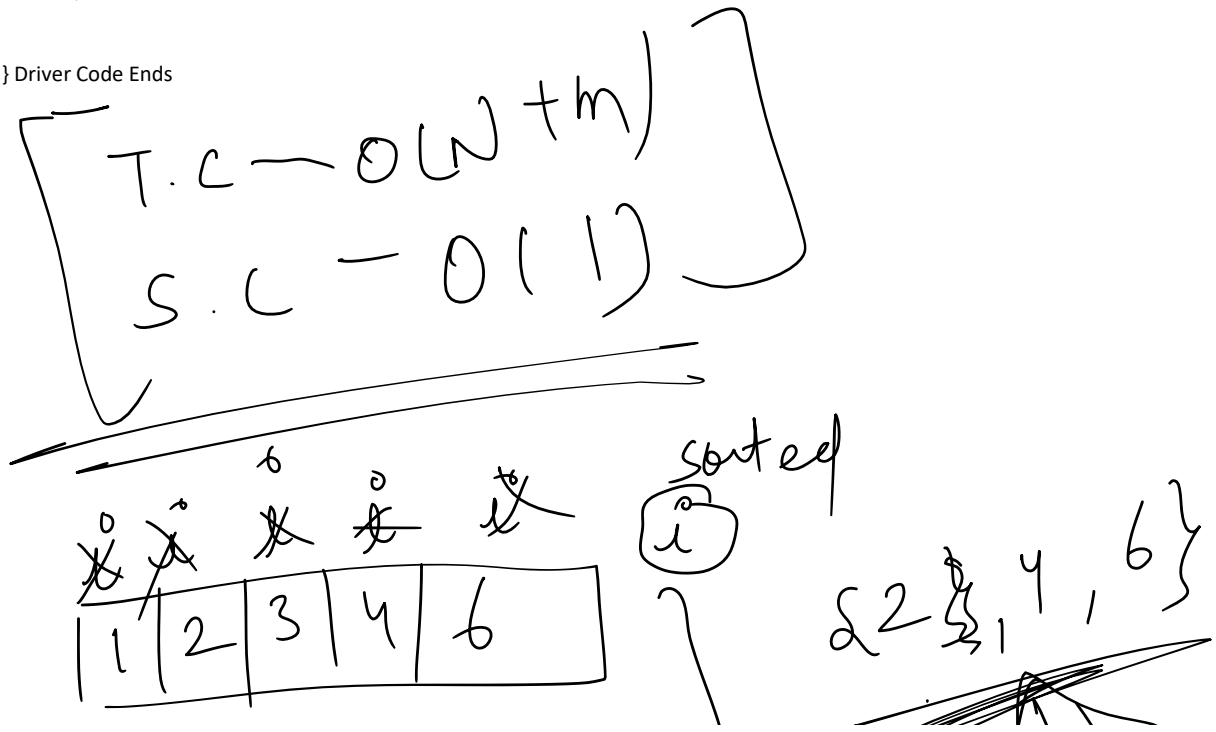

head
temp
1
2
3
1
temp
temp
temp
head
temp

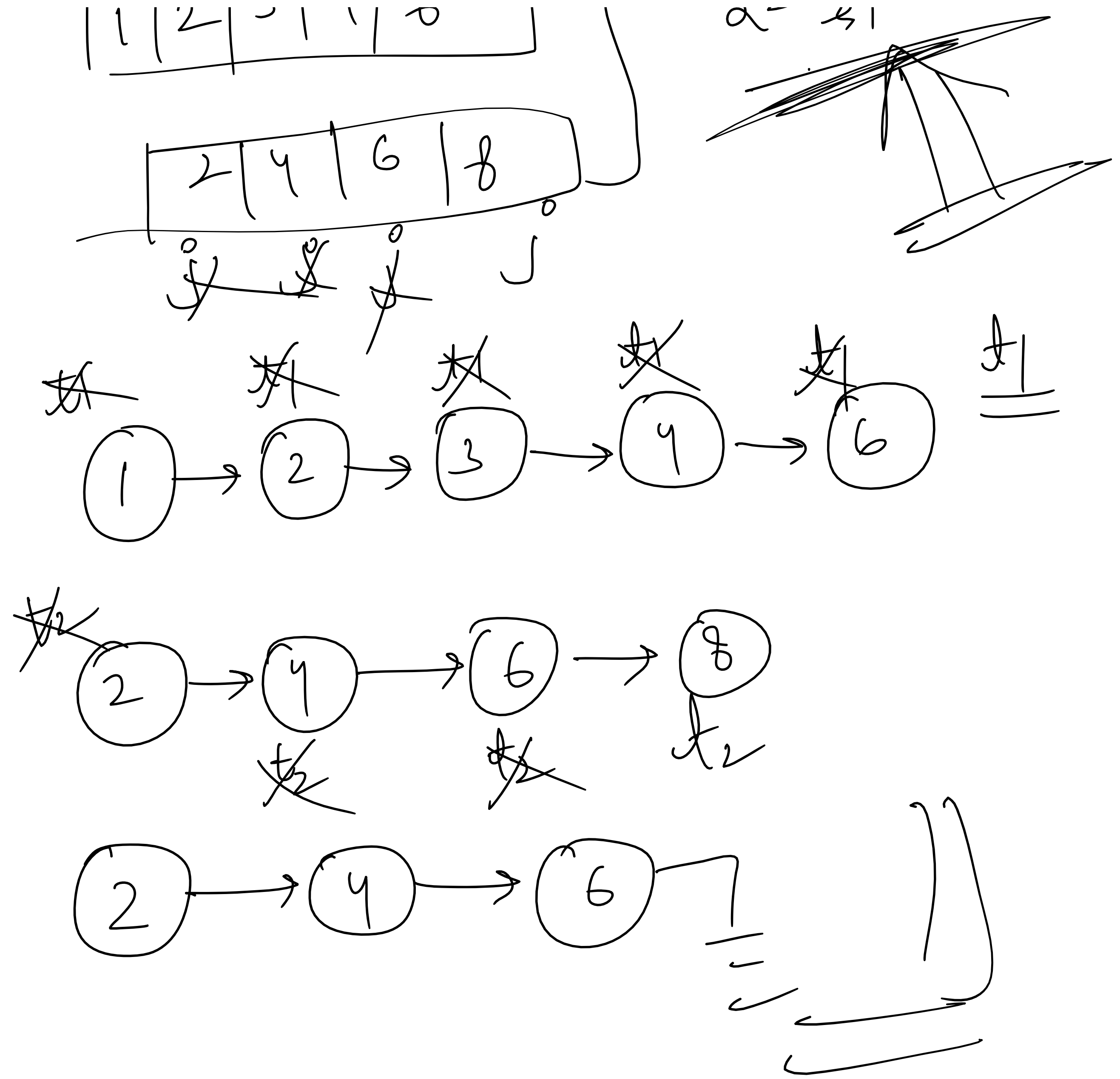


```
//{ Driver Code Starts
// driver

#include <bits/stdc++.h>
using namespace std;

/* Linked list Node */
struct Node {
    int data;
    struct Node* next;
    Node(int x) {
        data = x;
        next = NULL;
    }
};

struct Node* buildList(int size)
{
    int val;
    cin>> val;

    Node* head = new Node(val);
    Node* tail = head;

    for(int i=0; i<size-1; i++)
    {
        cin>> val;
        tail->next = new Node(val);
        tail = tail->next;
    }

    return head;
```

```
}

void printList(Node* n)
{
   while(n)
   {
      cout<< n->data << " ";
      n = n->next;
   }
   cout<< endl;
}


// } Driver Code Ends
/* node for linked list:

struct Node {
   int data;
   struct Node* next;
   Node(int x) {
      data = x;
      next = NULL;
   }
};

*/

class Solution
{
   public:
   //Function to add two numbers represented by linked list.
   Node * reverse(Node *head){
      Node *prev = NULL,*curr = head,*next = NULL,*tail = head;
     while(curr!=NULL){
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
     }
     head =prev;
     return head;
   }
   struct Node* addTwoLists(struct Node* first, struct Node* second)
   {
      // reverse both the linked lists
      Node * new_head1 = reverse(first);
      Node * new_head2 = reverse(second);

      Node *temp1 = new_head1,*temp2 = new_head2;
      int carry = 0;
      Node *head = NULL,*temp = NULL;

      while(temp1 != NULL && temp2 != NULL){
         int sum = temp1->data + temp2->data + carry;
         int value = sum%10;
         carry = sum/10;
         if(head == NULL){
            head = temp = new Node(value);
         }else{
            temp->next = new Node(value);
            temp = temp->next;
         }
         temp1 = temp1->next;
         temp2 = temp2->next;
      }
      if(temp1 == NULL){
         while(temp2 != NULL){
            int sum = temp2->data + carry;
            int value = sum%10;
            carry = sum/10;
            if(head == NULL){
               head = temp = new Node(value);
            }else{
```

```
            temp->next = new Node(value);
            temp = temp->next;
          }
          temp2 = temp2->next;
        }
      }else{
        while(temp1 != NULL){
          int sum = temp1->data + carry;
          int value = sum%10;
          carry = sum/10;
          if(head == NULL){
            head = temp = new Node(value);
          }else{
            temp->next = new Node(value);
            temp = temp->next;
          }
          temp1 = temp1->next;
        }
      }
      if(carry !=0){
        temp->next = new Node(carry);
      }
      return reverse(head);
      // return head;
    }
};


//{ Driver Code Starts.

int main()
{
   int t;
   cin>>t;
   while(t--)
   {
      int n, m;

      cin>>n;
      Node* first = buildList(n);

      cin>>m;
      Node* second = buildList(m);
      Solution ob;
      Node* res = ob.addTwoLists(first,second);
      printList(res);
   }
   return 0;
}

// } Driver Code Ends
```



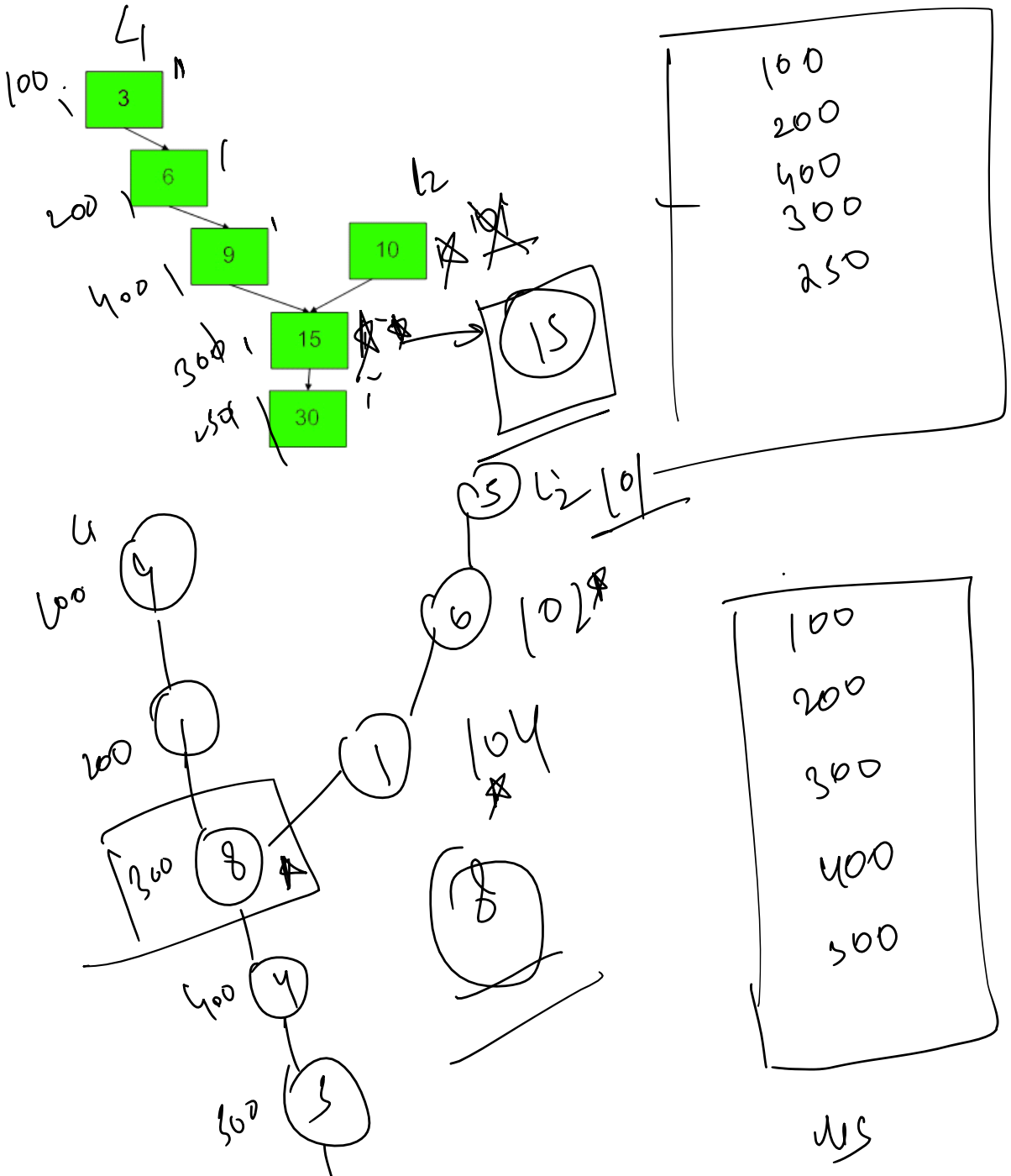


```
Node* findIntersection(Node* head1, Node* head2)
{
    Node * temp1 = head1,*temp2 = head2;
    Node *head = NULL,*temp = NULL;
    while(temp1 != NULL && temp2 != NULL){
        if(temp1->data < temp2->data){
            temp1 = temp1->next;
        }
        else if(temp2->data < temp1->data){
            temp2 = temp2->next;
        }else{
            if(head == NULL){
                head = temp = new Node(temp1->data);
            }else{
                temp->next = new Node(temp1->data);
                temp = temp->next;
            }
            temp1 = temp1->next;
            temp2 = temp2->next;
        }
    }
    return head;
}
```

T.C - O(min(N,M))

T.C - O(min(N, M))

S.C - O(min(N, M))

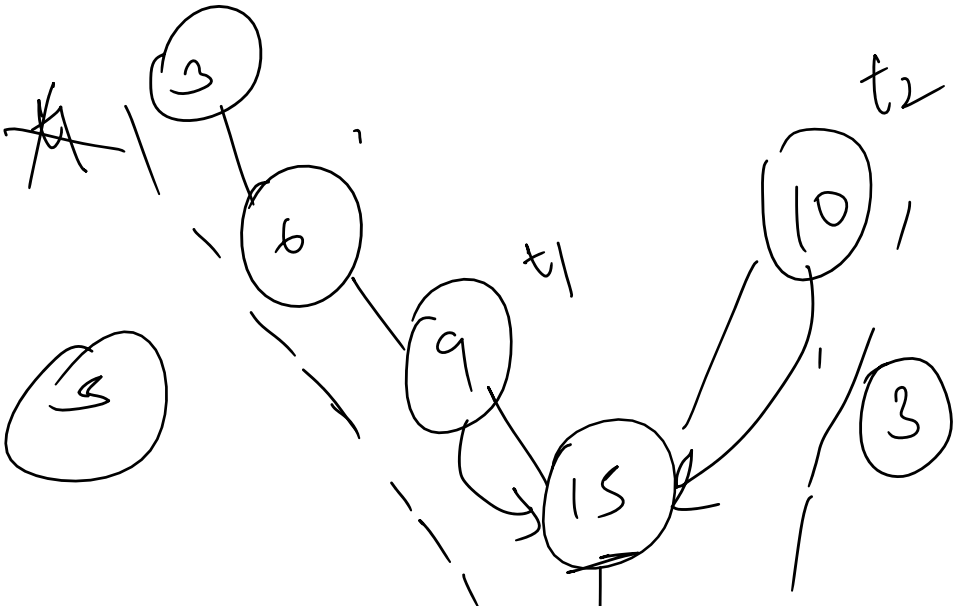


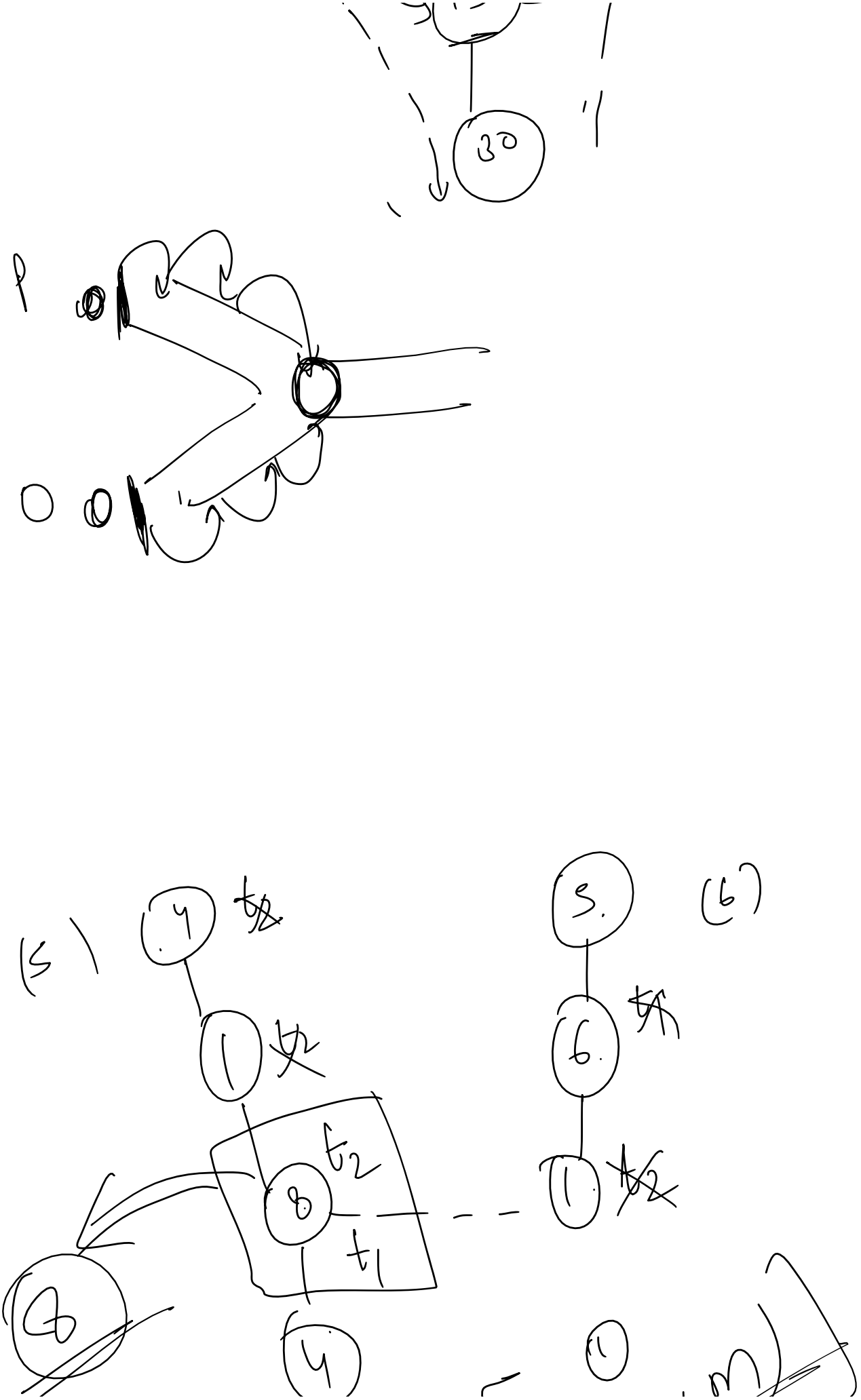
```
int intersectPoint(Node* head1, Node* head2)
{
    Node * temp1= head1;
    unordered_set<Node *> us;
    while(temp1 != NULL){
        us.insert(temp1);
        temp1 = temp1->next;
    }
    Node * temp2 = head2;
    while(!(us.find(temp2)!=us.end())){
        temp2 = temp2->next;
    }
    return temp2->data;
}
```

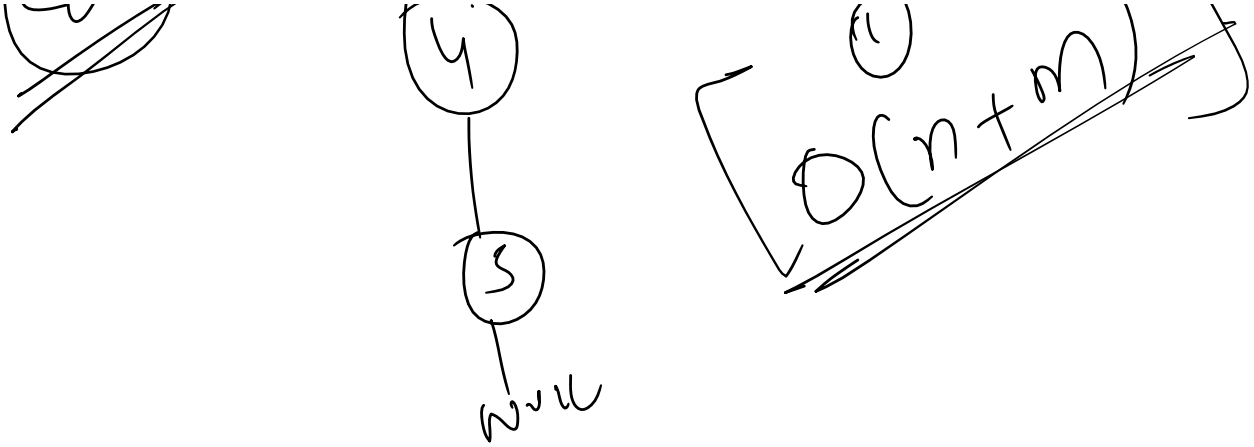
TLE

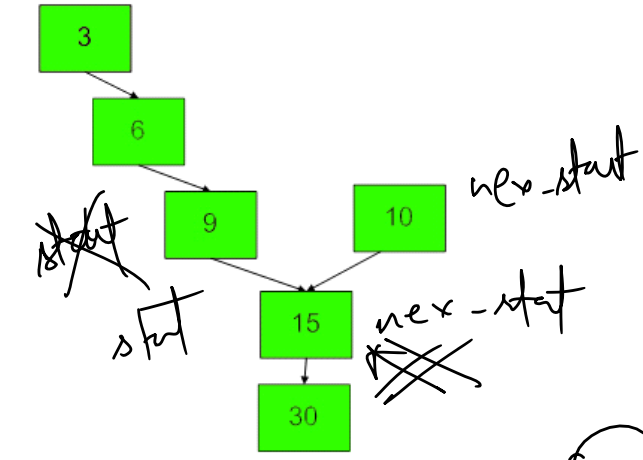
T.C – O(n+m)

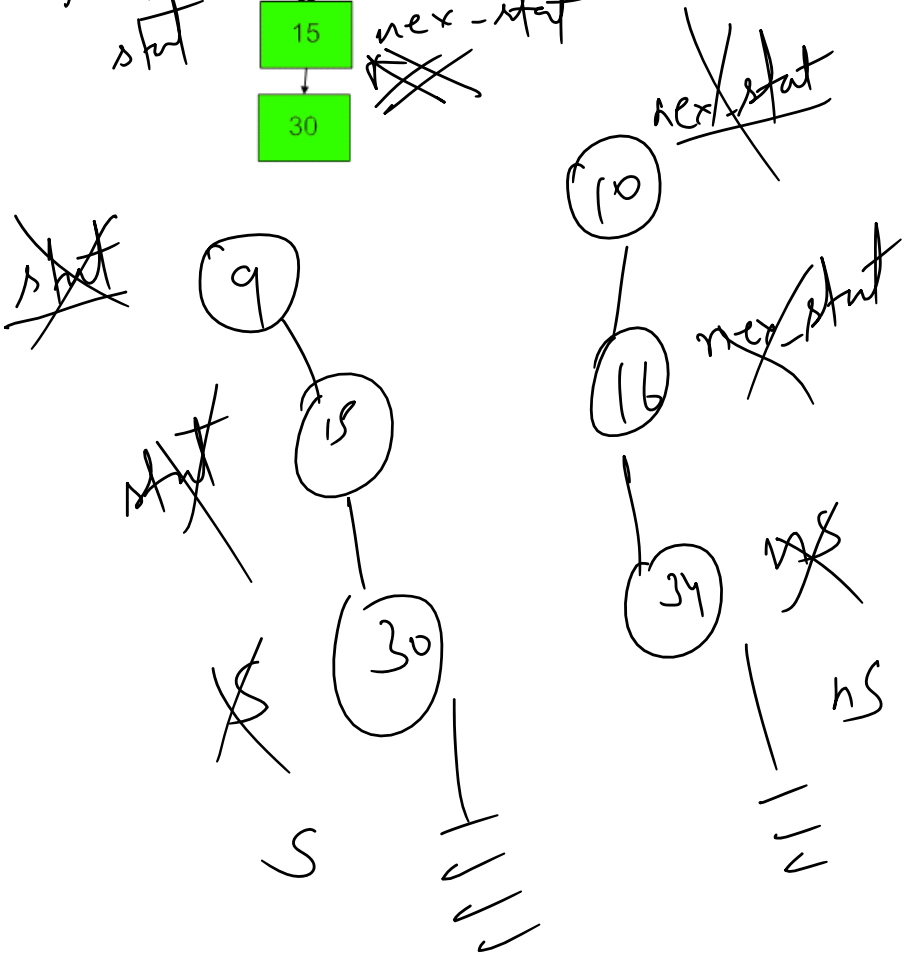
S.C – O(n)




30
t2
t1


O(n+m)
Null


3
6
9
10
15
30


10
16
34
9
15
30

```
int intersectPoint(Node* head1, Node* head2)
{
    int ll1  = 0,ll2 = 0;
    Node *temp1 = head1;
    while(temp1!=NULL){
        ll1++;
        temp1 = temp1->next;
    }
    Node *temp2 = head2;
    while(temp2!=NULL){
        ll2++;
        temp2 = temp2->next;
    }
    Node * start = NULL;
    Node * nex_start = NULL;
    int diff = 0;
    if(ll1 > ll2){
        start = head1;
        nex_start = head2;
        diff = ll1 - ll2;

    }else{
        start = head2;
        nex_start = head1;
        diff = ll2 - ll1;
    }
    while(diff!=0){
        diff--;
        start = start->next;
    }
    while(start != nex_start){
        start = start->next;
        nex_start = nex_start->next;
    }
    if(start == NULL) return -1;
    return start->data;
}
```

T.C- O(n+m)

S.C- O(1)