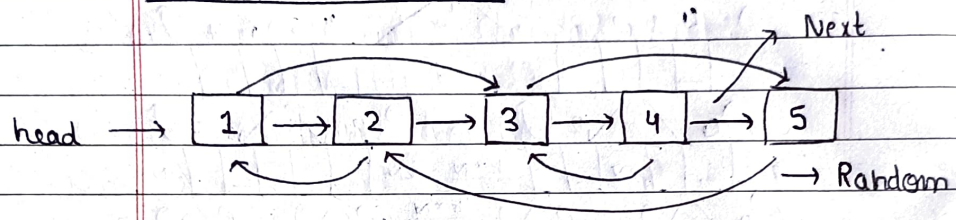
Date _____ Page _____

Day - 127

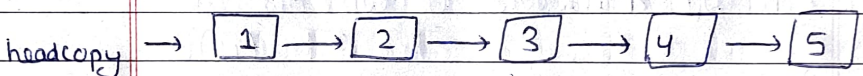
Linked List - II

* **Clone a Linked List:**



⇒ First, we will create a LL by using next pointer.

⇒ So, after that —



⇒ Our clone LL will looks like ⌐

⇒ Now, we have to assign random pointer ~~now~~

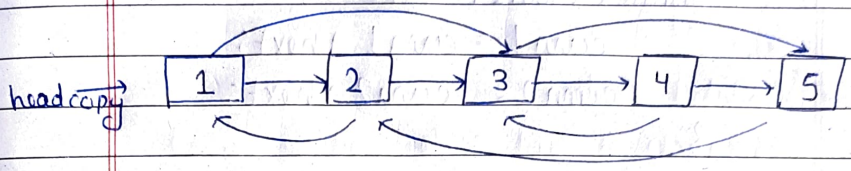
⇒ For that, we will have to find the random pointer going to which node.

⇒ So, we will calculate the position of the node by calculating the distance from the current node.

⇒ So, distance is same for original and clone.

Date _____ Page _____

⇒ So, after doing all the above steps — Our LL will looks like this —



Code

```
Node *headcopy = new Node(0);
Node * tailcopy = headcopy;
Node *temp = head;
while(temp){
    tailcopy -> next = new Node(temp -> data);
    tailcopy = tailcopy -> next;
    temp = temp -> next;
}
tailcopy = headcopy;
headcopy = headcopy -> next;
delete tailcopy;

tailcopy = headcopy;
temp = head;

while(temp){
    tailcopy -> arb = find(head, headcopy, temp ->
                           arb);
    tailcopy = tailcopy -> next;
    temp = temp -> next;
}
return headcopy;
```

Date _____ Page _____

```
Node * find ( Node * curr1 , Node * curr2,
              Node * x){ if(x== NULL) return NULL;
    while( curr1 != x){
        curr1 = curr1 → next;
        curr2 = curr2 → next;
    }
    return curr2;
}
```

⇒ This ~~qui~~ approach will have T.C. of — $O(n^2)$.

⇒ But we have to solve this question in $O(n)$ T.C.

⇒ Our T.C. increased due to the using of find function.

⇒ Suppose, if we can find the address of the arb. node in $O(1)$ time then our problem can be solved in $O(n)$ time.

⇒ New Code

Code is same until (temp = head)

Date _____ Page _____

```
unordered_map<Node*, Node*> m;
while(temp){
    m[temp] = tailcopy;
    temp = temp->next;
    tailcopy = tailcopy->next;
}
temp = head;
tailcopy = headcopy;

while(temp){
    tailcopy->arb = m[temp->arb];
    tailcopy = tailcopy->next;
    temp = temp->next;
}
return headcopy;
```

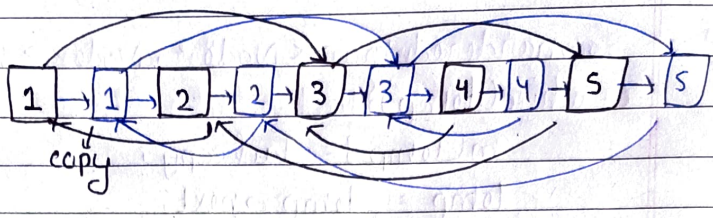
T.C. → O(n)
S.C. → O(n)

⇒ This is also not the most optimized approach.

⇒ If somehow if any node will know their copy node then without using any extra space, we can solve our problem.

⇒ So for this, we can make a link by shifting the nodes of copy LL to the right of every original LL.

Date _____ Page _____

⇒ 

Updated Code
(Creation of SLL code will be same)

```
⇒  Node * curr1 = head, * curr2 = headcopy;
   Node * front1, * front2;

   while(curr1){
      front1 = curr1->next;
      front2 = curr2->next;
      curr1->next = curr2;
      curr2->next = front1;
      curr1 = front1;
      curr2 = front2;
   }

   curr1 = head;
   curr2 = headcopy;
   while (curr1){
      curr2 = curr1->next;
if(curr2->arb){
      curr2->arb = curr1->arb->next; }
      curr1 = curr2->next;
   }
   curr1 = head;
```

Date: Page:

```
while( curr1->next){
    front1 = curr1->next;
    curr1->next = front1->next;
    curr1 = front1;
}
return headcopy;
```