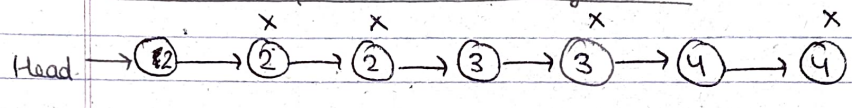
Date: / /
Page:

Day - 120

Linked List - 6

* **Remove duplicate element from list:**



⇒ We have to delete the duplicate & return the new LL.

⇒ We can create an array and insert the element without duplicates.

⇒ And then again traverse the linked list & insert the array elements.

2	3	4

⇒ But this approach will require extra memory.

Code

```
vector<int> ans;
ans.push_back(head->data);
Node *curr = head->next;
while(curr){
    if(ans[ans.size()-1] != curr->data)
        ans.push_back(curr->data);
    curr = curr->next;
}
curr = head;
```

```
int index = 0;
while (index < ans.size()){
        curr → data = ans [index++];
        curr = curr → next;
}
```

~~curr~~

```
int size = ans.size()-1;
curr = head;
while(size--){
    curr = curr → next;
}
curr → next = NULL;
return head;
```

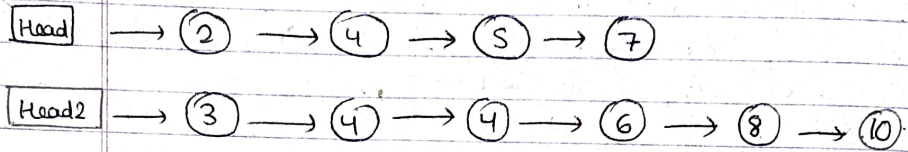
⇒ But now, we have to solve this question inplace.
⇒ We will take two pointers prev & curr.
⇒ when the value of curr & prev pointer is same then we will delete the curr pointer.

Code

```
Node *curr = head → next; *prev = head;
while (curr){
    if ( curr → data == prev → data) {
        prev → next = curr → next;
        delete curr;
        curr = prev → next;
    } else {
        prev = prev → next;
        curr = curr → next;
    }
}
return head;
```

Date: / /
Page:

* Merge two sorted List:



⇒ In the first approach, we can create a new linked list by using two pointer on the LL1 & LL2.

⇒ And then check, the smaller comes first and in this way, we will create our new LL.

⇒ But this approach will take extra memory.

⇒ In the second approach, we don't create any new node.

⇒ Instead, we will use the created nodes that given in the question.

⇒ ~~Code~~ In the code part, we will create a head and node so that our code will be shorter.

Code

```
Node *head = new Node(0);
Node *tail = head;
```

*

head

⇒

```
while(head1 && head2){
    if(head1 → data <= head2 → data){
        tail → next = head1;
        head1 = head1.next;
        tail = tail. → next;
        tail → next = NULL;
    } else{
        tail → next = head2;
        head2 = head2.next;
        tail = tail → next;
        tail → next = NULL;
    }
}
if(head1){
    tail → next = head1;
} else
    tail → next = head2;
}
return head
tail = head;
head = head → next;
delete tail;
return head;
```

* Sort a list which contain 0, 1, 2:

curr ↓

head → (0) → (1) → (0) → (2) → (2) → (0) → (1)

⇒ First, we will count the no. of 0, 1 & 2,

count 0 = 3

count 1 = 2

count 2 = 2

Date: / /
Page:

⇒ Now, we will chang the values an update the values according to the count.

Code:

```
Node * curr = head;
int count0 = 0 , count1 = 0, count2 = 0;
while( curr ){
    if(curr → data == 0) count0++;
    else if (curr → data == 1) count1++;
    else count2++;
    curr = curr → next;
}
curr = head;
while( count0--){
   curr → data = 0;
   curr = curr → next;
}
while( count1--){
   curr → data = 1;
   curr = curr → next;
}
while( count2-- ) {
   curr → data = 2;
   curr = curr → next;
}
return head;
```

Date: / /
Page:

=) If the condition is that we can't change the value of nodes then —

=) First we create three pointers —

head 0 : ⓪ → ⓒ → ⓒ

head 1 : ① → ①

head 2 : ② → ②

=) After that we will attach ~~the~~ the no. with their respective head like 0 will attach with head 0.