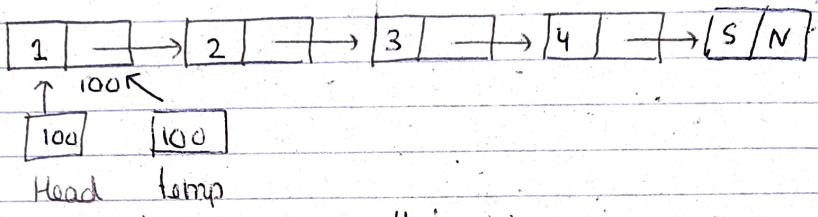
Date : / /
Page:

Day - 115

Linked List - 3

* Reverse LL;



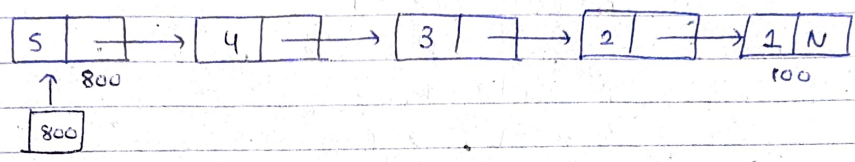
⇒ We want to reverse this LL.

⇒ We can do that we can traverse the LL & store the data in arr then again traverse the LL & change the value by the value of array starting from end.

⇒

```
vector <~~ans~~ int> ans;
ListNode *temp = head;
while( temp != NULL){
    ans.push_back(temp->val);
    temp = temp->next;
}
int i= ans.size()-1;
temp = head;
while (temp){
   temp->val = ans[i];
   i--;
   temp = temp->next;
}
```

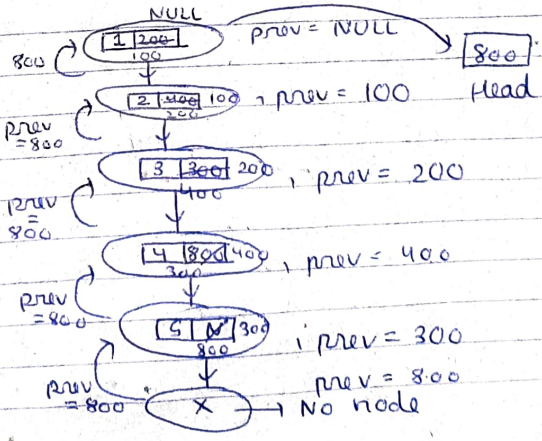
⇒ But we want to reverse the whole LL like as —



⇒ We will take curr & prev pointer.

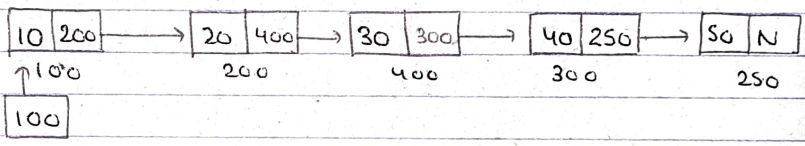
```
ListNode *curr = head, *prev = NULL, *fut = NULL;
while(curr){
    fut = curr->next;
    curr->next = prev;
    curr = fut;
    prev = curr;
    curr = fut;
}
head = prev;
```

⇒ So, now, we have to solve this problem by using recursion.



```
ListNode* Reverse (ListNode* curr, ListNode* prev){
    if (curr == NULL){
        return prev;
    }; ListNode* fut = curr->next;
    curr->next = prev;
    return Reverse(fut, curr);
}
```

* Middle of Linked List:



⇒ Return the middle of the LL.

⇒ First, we will count the no. of nodes then divide the count by 2.

⇒ After that, we will jump to that node by traversing.

⇒ Now, our first task is to count the nodes. For that —

```
int count = 0;
ListNode* temp = head;
while(temp != NULL){
    count++;
    temp = temp->next;
}
```

```
count /= 2;
temp = head;
while(count--){
    temp = temp->next;
}
return temp;
```

⇒ Now, can we solve this problem in 1 traversal?

⇒ Yes, we can solve it by using a ~~new~~ concept known as fast & slow pointer.

⇒ It is a very imp. concept of Linked List.

⇒ In this, we use slow & fast pointers.

⇒ fast pointer is double in speed of slow pointer.

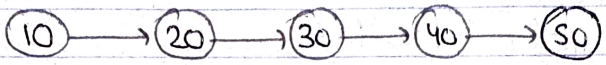
⇒ So in our question, we put two pointers at the starting.

⇒ Now, slow pointer will do 1 jump at a time while fast pointer do 2 jumps at a time.

⇒
```
ListNode *slow = head, *fast = head;
while(fast!=NULL && fast->next!=NULL){
    slow = slow->next;
    fast = fast->next->next;
}
return slow;
```

Date: / /
Page:

* **Rotate List**



⇒ So, we will have a value of k and that time we have rotate our list.

⇒ Suppose k = 2,

⇒ So, our task is to put 'k' no. of nodes from last and put it in the front as starting.