Q) Rotate a Linked List

Code:

//first we find point using curr .then after we get we wiil asine it to temp. and traver curr to end and curr->next to head .and now change head ;///

Node\* rotate(Node\* head, int k)

{

// Your code here

Node\* curr=head;

int i=1;

if(k==0)

{

return head;

}

while(i<k && curr!=NULL)

{

curr=curr->next;

i++;

}

if(curr==NULL)

{

return head;

}

Node\* temp1=curr;

while(curr->next!=NULL)

{

curr=curr->next;

}

curr->next=head;

head=temp1->next;

temp1->next=NULL;

return head;

}

};

2) **Reverse a Linked List in groups of given size | Set 1**

struct node \*reverse (struct node \*head, int k)

{

// Complete this method

node\* curr=head;

node\* next=NULL;

node\* prev=NULL;

int count=0;

if(head==NULL)

{

return head;

}

while(count<k && curr!=NULL)

{

next=curr->next;

curr->next=prev;

prev=curr;

curr=next;

count++;

}

if(next!=NULL)

{

head->next=reverse(next,k);

}

return prev;

}

};

3) **Given a linked list of 0s, 1s and 2s, sort it**

**Code:**

class Solution

{

public:

//Function to sort a linked list of 0s, 1s and 2s.

Node\* segregate(Node \*head) {

// Add code here

int count[]={0,0,0};//to make a count on number of 0s,1s,2s ;

Node\* curr=head;

while(curr!=NULL)

{

count[curr->data]+=1;

curr=curr->next;

}

//from above step we will get count of each nuimber

curr=head;

int i=0;

while(curr!=NULL)

{

if(count[i]==0)

{

i++;//now we check if count became 0 or not if it becomes 0 we will increment i

}

else

{

curr->data=i;

count[i]--;

curr=curr->next;

}

}

return head;

}

};

4) **Odd Even Linked List**

**Code**

ListNode\* oddEvenList(ListNode\* head)

{

if(head==NULL)

{

return head;

}

ListNode\* even=head->next;

ListNode\* odd=head;

ListNode\* evenhead=even;

while(even!=NULL && even->next!=NULL)

{

odd->next=even->next;

odd=odd->next;

even->next=odd->next;

even=even->next;

}

odd->next=evenhead;

return head;

}

};