Delete Kth Node

Iterate and find a node just before kth node if it is head then make head→next as new head.

If it is middle then find just previous node and point to deleted element next node. If it is last node then just point its previous to NULL.

```
/*
Following is the class structure of the Node class:

class Node
{
public:
    int data;
    Node *next;
    Node()
    {
        this->data = 0;
        next = NULL;
    }
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }
    Node(int data, Node* next)
    {
        this->data = data;
        this->next = next;
    }
};
*/

Node* removeKthNode(Node* head, int K)
{
    // Write your code here.
    Node* temp=head;
    int cnt=0;
    while(temp!=NULL)
    {
        cnt++;
        temp=temp->next;
    }
    temp=head;
    if(K==cnt)
    {
        head=head->next;
```

```
        delete(temp);
        return head;
    }
    int ind=cnt-K;
    cnt=0;
    while(temp!=NULL)
    {
        cnt++;
        if(cnt==ind)
        {
            if(K==1)
            {
                temp->next=NULL;
                return head;
            }
            temp->next=temp->next->next;
            return head;
        }
        temp=temp->next;
    }
    return head;
}
```

- Time Complexity : O(N)+ O(N)
- Space Complexity : O(1)

Optimised Approach :

Using two pointer. First we initialise the fast pointer to n from starting we know if length is l then l-n from end will give us the required pointer so we keep our fast pointer at n initially from start and then we start moving slow pointer once fast reaches end we know our slow will point to l-n because when fast was at n from start then we started moving slow so it must be n positions back from fast which means if fast is pointing to end then slow will be pointing to n positions back from end but to delete kth node we need a node just before it so the while condition will be when fast→next≠NULL instead of fast≠NULL.

```
/*
Following is the class structure of the Node class:

class Node
{
public:
```

```
    int data;
    Node *next;
    Node()
    {
        this->data = 0;
        next = NULL;
    }
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }
    Node(int data, Node* next)
    {
        this->data = data;
        this->next = next;
    }
};
*/

Node* removeKthNode(Node* head, int K)
{
    // Write your code here.
    Node* dummy=new Node();
    dummy->next=head;
    Node* fast=dummy;
    Node* slow=dummy;
    int n=K;
    for(int i=1;i<=K;i++)
    {
        fast=fast->next;
    }
    while(fast->next!=NULL)
    {
        slow=slow->next;
        fast=fast->next;
    }
    slow->next=slow->next->next;
    return dummy->next;
}
```

- Time Complexity : O(N)
- Space Complexity : O(1)