LINKED LIST Common Problems

By Tahmid

```
Node* reversedLinkedList(Node* head){
    if(head==nullptr)return nullptr;
    if(head->next==nullptr)return head;
    Node* curr=head;
    Node* post=nullptr;
    Node* prev=nullptr;
    while(curr){
    post=curr->next;
    curr->next=prev;
    prev=curr;
    curr=post;
    }
    return prev;
}
```

FLOYDS ALGORITHM

```
bool ifCycle(Node* head){
    if(head==nullptr)return false;
    if(head->next==nullptr)return false;
    Node* fast=head;
    Node* slow=head;

    while(fast && fast->next){
        fast=fast->next->next;
        slow=slow->next;
        if(fast==slow)return true;
    }
    return false;
}
```

```
Node* middleOfList(Node* head){
    Node* slow=head;
    Node* fast=head;
    while(fast && fast->next){
        fast=fast->next->next;
        slow=slow->next;

    }
    return slow;
}
```

Removal of Nth Node

```
Node* removeNnode(Node* head, int n){
    int nodecount=0;
    Node* curr=head;
    while(curr){
        nodecount++;
        curr=curr->next;
    }
    curr=head;
    if(nodecount<n) return;
    if(nodecount==n){
        Node* temp=head->next;
        delete head;
        head=temp;
        return head;

    }
    for(int i=0;i<(nodecount-n)-1;i++){
        curr=curr->next;
    }
    Node* prev=curr;
    curr=curr->next;
    Node* post=curr->next;
    prev->next=post;
    delete curr;
    return head;

}
```

Merging Sorted

```
Node* MergedList(Node* head1, Node* head2){
    Node* curr1=head1;
    Node* curr2=head2;
    Node* head=nullptr;
    if(curr1->data>curr2->data){
        head->data=curr2->data;
        curr2=curr2->next;
    }
    else {
        head->data=curr1->data;
        curr1=curr1->next;
    }
    Node* curr=head;
    while(curr1 || curr2){
        if(curr1&& curr2){
        if(curr1->data>curr2->data){
            curr->data=curr2->data;
            curr2=curr2->next;
        }
        else{
            curr->data=curr1->data;
            curr1=curr1->next;
        }

    }
    else if(curr1){
        curr->data=curr1->data;
    }
    else curr->data=curr2->data;
    curr=curr->next;



    }
    return head;
}
```

Intersection of Sorted SLL

```
Node* intersection(Node* head1,Node* head2){
    Node* dummy=new Node(-1);
    Node* curr=dummy;
    Node* curr1=head1;
    Node* curr2=head2;
    while(curr1 && curr2){
        if(curr1->data==curr2->data){
            curr->next = new Node(curr1->data);
            curr1=curr1->next;
            curr2=curr2->next;
            curr=curr->next;
        }
        else if(curr1->data>curr2->data){
            curr2=curr2->next;
        }
        else curr1=curr1->next;
    }
    return dummy->next;
}
```