Reverse a Linked List:

```
class Solution {
public:
    ListNode* reverseList(ListNode* head) {
        ListNode*prev=NULL;
        ListNode*curr=head;
        ListNode*next=NULL;
        while(curr!=NULL){
            next=curr->next;
            curr->next=prev;
            prev=curr;
            curr=next;
        }
        return prev;
    }
};
```

Time:O(No. Of Nodes)
Space:O(1)

Middle Node of the Linked List:

```
class Solution {
public:
    typedef ListNode Node;
    ListNode* middleNode(ListNode* head) {
        Node*slow=head;
        Node*fast=head;
        while(fast!=NULL&&fast->next!=NULL){
            slow=slow->next;
            fast=fast->next->next;
        }
        return slow;
    }
};
```

Time:(n)
Space:o(1)

Palindrome Linked List:

```
class Solution {
public:
    typedef ListNode Node;
```

```
    ListNode* reverseList(ListNode* head) {
        ListNode*prev=NULL;
        ListNode*curr=head;
        ListNode*next=NULL;
        while(curr!=NULL){
            next=curr->next;
            curr->next=prev;
            prev=curr;
            curr=next;
        }
        return prev;
    }

    ListNode* middleNode(ListNode* head) {
        Node*slow=head;
        Node*fast=head;
        while(fast->next!=NULL&&fast->next->next!=NULL){
            slow=slow->next;
            fast=fast->next->next;
        }
        return slow;
    }
    bool isPalindrome(ListNode* head) {
        Node*midNode=middleNode(head);
        Node*newHead=midNode->next;
        midNode->next=NULL;
        Node*nHead=reverseList(newHead);

        Node*p1=head;
        Node*p2=nHead;
        while(p1!=NULL&&p2!=NULL){
            if(p1->val!=p2->val){
                return false;
            }
            p1=p1->next;
            p2=p2->next;
        }
        return true;

    }
};
```

Time:O(n)+O(n/2)+(n/2+n/2)
Space:O(1)

Swapping Nodes in a Linked List:

```
class Solution {
public:
    typedef ListNode Node;
    int countNodes(Node*head){
        int cnt=0;
        Node*curr=head;
        while(curr!=NULL){
            cnt++;
            curr=curr->next;
        }
        return cnt;
    }
    ListNode* swapNodes(ListNode* head, int k) {
        Node*p1=head;
        Node*p2=head;
        int cnt=countNodes(head);
        for(int i=1;i<=k-1;i++){
            p1=p1->next;
        }
         for(int i=1;i<=cnt-k;i++){
            p2=p2->next;
        }
        cout<<p1->val<<","<<p2->val<<endl;
        swap(p1->val,p2->val);
        return head;
    }
};
```

Time:O(n)

Space:O(1)