1. **Reverse a linked list (Asked in adobe, amazon, de shaw, goldman sachs, intuit, microsoft, paytm, qualcomm, samsung, walmart, vmware)**

Code :

```
struct Node* reverseList(struct Node *head)
  {
       if(head==NULL)
       return NULL;

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

2. **Middle of the Linked List (asked in adobe, amazon, flipkart, microsoft, morgan stanley, samsung, sap labs)**

Code using two traversals :

```
ListNode* middleNode(ListNode* head) {
        int n=0;
        ListNode* curr=head;
       while(curr!=NULL)
       {
            n++;
            curr=curr->next;
       }
       int mid=n/2;
       curr=head;
       while(mid>0)
       {
            mid--;
           curr=curr->next;
       }
       return curr;
   }
```

Code using one traversal (fast and slow pointer) :

```
ListNode* middleNode(ListNode* head) {
    ListNode* slow=head;
    ListNode* fast=head;

    while(fast!=NULL and fast->next!=NULL)
    {
        slow=slow->next;
        fast=fast->next->next;
    }
    return slow;
}
```

3. **Check if Linked List is Palindrome (asked in adobe, amazon, microsoft)**

Code using the approach of reversing the 2nd half of the linkedlist

```
class Solution{
  Public:

    //Function to check whether the list is palindrome.

    bool isPalindrome(Node *head)
    {
        //Your code here
        //  FINDING THE MIDDLE

        Node* slow=head;
        Node* fast=head;

        while(fast!=NULL and fast->next!=NULL)
        {
            slow=slow->next;
            fast=fast->next->next;
        }

        // reverse the 2nd half
        Node* prev=NULL;
        Node* curr=slow;


        while(curr!=NULL)
        {
             Node* nextNode=curr->next;
             curr->next=prev;
             prev=curr;
             curr=nextNode;
        }
```

```
        // check for palindrome

        while(prev!=NULL)
        {
            if(head->data!=prev->data)
            return false;

            head=head->next;
            prev=prev->next;
        }
        return true;

    }

};
```

Approach 2 code using recursion (note you can also use stack instead of recursion)

```
Node* curr;
  Node* temp;
  bool rec(Node* curr)
  {
        if(curr==NULL)
        return true;
        bool ans;
        ans= rec(curr->next) and temp->data==curr->data;
        temp=temp->next;
        return ans;
  }
  bool isPalindrome(Node *head)
  {
     curr=head;
     temp=head;
       return rec(curr);
  }
```

4. **Merge two sorted linked lists (asked in amazon, flipkart, microsoft)**

Code using recursion approach :

```
Node* sortedMerge(Node* head1, Node* head2)
{
   // code here
       if(head1==NULL)
       return head2;
       if(head2==NULL)
       return head1;

       if(head1->data<head2->data)
       {
            head1->next=sortedMerge(head1->next,head2);
            return head1;
       }
       else
       {
           head2->next=sortedMerge(head1,head2->next);
           return head2;
       }
}
```

Code using iterative approach :

```
Node* sortedMerge(Node* head1, Node* head2)
{

        if(head1==NULL)
        return head2;

        if(head2==NULL)
        return head1;

        Node *res,*tail;

        if(head1->data<head2->data)
        {
              res=head1;
              tail=head1;
              head1=head1->next;
        }

        else
        {
               res=head2;
               tail=head2;
               head2=head2->next;
        }

         while(head1!=NULL and head2!=NULL)
         {

               if(head1->data<head2->data)
               {

                    tail->next=head1;
                    tail=head1;
```

```
            head1=head1->next;

        }

        else
        {

            tail->next=head2;
            tail=head2;
            head2=head2->next;

        }

}

if(head1!=NULL)
tail->next=head1;

if(head2!=NULL)
tail->next=head2;

return res;

}
```

5. Move all zeros to the front of the linked list

Code :

```
void moveZeroes(struct Node **head)
{
   //Your code here
      Node* prev=*head;
      Node* curr=(*head)->next;
      while(curr!=NULL)
      {
            if(curr->data!=0)
            {
                prev=curr;
                curr=curr->next;
            }
            else
            {
                  prev->next=curr->next;
                  curr->next=*head;
                  *head=curr;
                  curr=prev->next;
            }
      }
}
```