**Day-5**

**Problem1**- [**876. Middle of the Linked List**](https://leetcode.com/problems/middle-of-the-linked-list/)

Solution-

class Solution {

public:

    ListNode\* middleNode(ListNode\* head) {

       ListNode\* slow=head,\*fast=head;

       while(fast && fast->next)

       {

           slow=slow->next;

           fast=fast->next->next;

       }

       return slow;

    }

};

**Problem2**- [**141. Linked List Cycle**](https://leetcode.com/problems/linked-list-cycle/)

Solution-

class Solution {

public:

    bool hasCycle(ListNode \*head) {

        ListNode\* f=head;

        ListNode\* s=head;

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

        {

            f=f->next->next;

            s=s->next;

            if(f==s)

            return true;

        }

        return false;

    }

};

**Problem 3**- [**2. Add Two Numbers**](https://leetcode.com/problems/add-two-numbers/)

class Solution {

public:

    ListNode\* addTwoNumbers(ListNode\* l1, ListNode\* l2) {

        ListNode \* nya=new ListNode();

        ListNode \*temp=nya;

        int carry=0;

        while(l1!=NULL || l2!=NULL||carry)

        { int sum=0;

            if(l1!=NULL)

            {

                sum+=l1->val;

                l1=l1->next;

            }

            if(l2!=NULL)

            {

                sum+=l2->val;

                l2=l2->next;

            }

            sum+=carry;

            carry=sum/10;

            ListNode \*node=new ListNode(sum%10);

            temp->next=node;

            temp=temp->next;

        }

        return nya->next;

    }

};

**Problem 4**- [**206. Reverse Linked List**](https://leetcode.com/problems/reverse-linked-list/)

class Solution {

public:

    ListNode\* reverseList(ListNode\* head) {

        ListNode \*temp = head;

        vector<int> v;

         if (head == nullptr) {

            return nullptr;

             }

        while (temp != nullptr) {

            v.push\_back(temp->val);

            temp = temp->next;

        }

         if (v.empty()) {

            return nullptr;

        }

        int m = v.size();

        reverse(v.begin(), v.end());

        ListNode \*

        curr = new ListNode(v[0]);

        ListNode \*ptr = curr;

        for (int i = 1; i < m; i++) {

            ListNode \*newNode = new ListNode(v[i]);

            ptr->next = newNode;

            ptr = newNode;

        }

        return curr;

    }

};