Delete the middle node of a linked list –

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     ListNode \*next;

 \*     ListNode() : val(0), next(nullptr) {}

 \*     ListNode(int x) : val(x), next(nullptr) {}

 \*     ListNode(int x, ListNode \*next) : val(x), next(next) {}

 \* };

 \*/

class Solution {

public:

    ListNode\* deleteMiddle(ListNode\* head) {

        ListNode\* first = head;

        ListNode\* second = head;

        ListNode\* prev = head;

        long long count = 0;

        if(head->next==NULL)

        return NULL;

        else{

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

        {

            first = first->next->next;

            second = second->next;

            if(count>0)

            prev=prev->next;

            count++;

        }

        prev->next = second->next;

        second->next = NULL;

        return head;

        }

    }

};

Merge two sorted lists

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     ListNode \*next;

 \*     ListNode() : val(0), next(nullptr) {}

 \*     ListNode(int x) : val(x), next(nullptr) {}

 \*     ListNode(int x, ListNode \*next) : val(x), next(next) {}

 \* };

 \*/

class Solution {

public:

        ListNode\* mergeTwoLists(ListNode\* list1, ListNode\* list2) {

        ListNode\* head = NULL;

        ListNode\* temp = NULL;

        if(list1 == NULL)

            return list2;

            else if(list2 == NULL)

            return list1;

        else if(list1->val <= list2->val)

        {

            temp = list1;

            head = temp;

            list1 = list1->next;

            while(list1!=NULL && list2!=NULL)

            {

                if(list1->val <= list2->val)

                {

                    temp->next = list1;

                    list1 = list1->next;

                    temp = temp->next;

                }

                else

                {

                    temp->next = list2;;

                    list2 = list2->next;

                    temp = temp->next;

                }

            }

            if(list1 == NULL)

            temp->next = list2;

            else if(list2 == NULL)

            temp->next = list1;

            return head;

        }

        else

        {

            temp = list2;

            head = temp;

            list2 = list2->next;

            while(list1!=NULL && list2!=NULL)

            {

                if(list1->val <= list2->val)

                {

                    temp->next = list1;

                    list1 = list1->next;

                    temp = temp->next;

                }

                else

                {

                    temp->next = list2;;

                    list2 = list2->next;

                    temp = temp->next;

                }

            }

            if(list1 == NULL)

            temp->next = list2;

            else if(list2 == NULL)

            temp->next = list1;

            return head;

        }

    }

};

Next greater node

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     ListNode \*next;

 \*     ListNode() : val(0), next(nullptr) {}

 \*     ListNode(int x) : val(x), next(nullptr) {}

 \*     ListNode(int x, ListNode \*next) : val(x), next(next) {}

 \* };

 \*/

class Solution {

public:

    vector<int> nextLargerNodes(ListNode\* head) {

        vector<int> nextarr;

        while(head != NULL){

            nextarr.push\_back(head->val);

            head = head->next;

        }

        stack<int> st;

        st.push(0);

        for(int i=nextarr.size()-1; i>=0; i--){

            int curr = nextarr[i];

            while(st.top() != 0 && curr >= st.top()){

                st.pop();

            }

            nextarr[i] = st.top();

            st.push(curr);

        }

        return nextarr;

    }

};

Maximum twin sum of a linked list

/\*\*

 \* Definition for singly-linked list.

 \* struct ListNode {

 \*     int val;

 \*     ListNode \*next;

 \*     ListNode() : val(0), next(nullptr) {}

 \*     ListNode(int x) : val(x), next(nullptr) {}

 \*     ListNode(int x, ListNode \*next) : val(x), next(next) {}

 \* };

 \*/

class Solution {

public:

    vector<int> nextLargerNodes(ListNode\* head) {

        vector<int> nextarr;

        while(head != NULL){

            nextarr.push\_back(head->val);

            head = head->next;

        }

        stack<int> st;

        st.push(0);

        for(int i=nextarr.size()-1; i>=0; i--){

            int curr = nextarr[i];

            while(st.top() != 0 && curr >= st.top()){

                st.pop();

            }

            nextarr[i] = st.top();

            st.push(curr);

        }

        return nextarr;

    }

};