**WORKSHEET 4**

**Student Name: Priyansh Tyagi**

**UID: 20BCS5670**

**Branch: CSE**

**Section: 903-A MM (DWWC- 43)**

# Que-1: [Remove Duplicates from Sorted List](https://leetcode.com/problems/remove-duplicates-from-sorted-list/description/)

**Code:**

class Solution { public:

ListNode\* deleteDuplicates(ListNode\* head) { if(head==nullptr)

return nullptr; ListNode \*temp=head;

while(temp!=nullptr && temp->next!=nullptr){ if(temp->val==temp->next->val){

temp->next=temp->next->next;

}else

temp=temp->next;

}

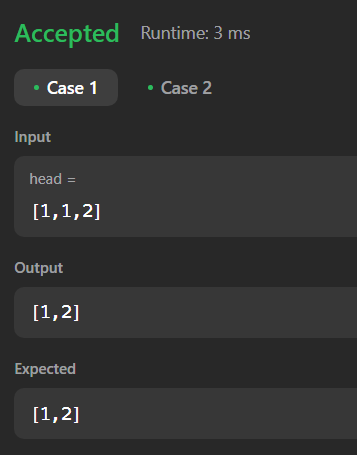
return head;

}

};

# Output:



**Que-2:** [**Palindrome Linked List**](https://leetcode.com/problems/palindrome-linked-list/description/)

# Code:

class Solution { public:

ListNode\* calmid(ListNode\* head){ ListNode\* slow=head; ListNode\* fast=head;

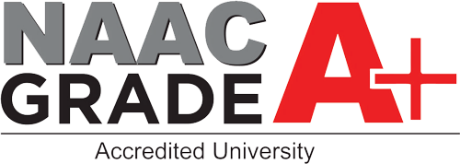
while(fast!=nullptr && fast->next!=nullptr){ slow=slow->next;

fast=fast->next->next;

}

return slow;

}



ListNode\* reverse(ListNode\* head){ ListNode\* curr=head;

ListNode \*temp=nullptr; ListNode\* prev=nullptr; while(curr!=nullptr){

temp=curr->next; curr->next=prev; prev=curr; curr=temp;

}

return prev;

}

bool isPalindrome(ListNode\* head) { ListNode\* p1=head;

ListNode\* mid=calmid(head); ListNode\* p2=reverse(mid); while(p1!=nullptr && p2!=nullptr){

if(p1->val!=p2->val) return false;

p1=p1->next; p2=p2->next;

}

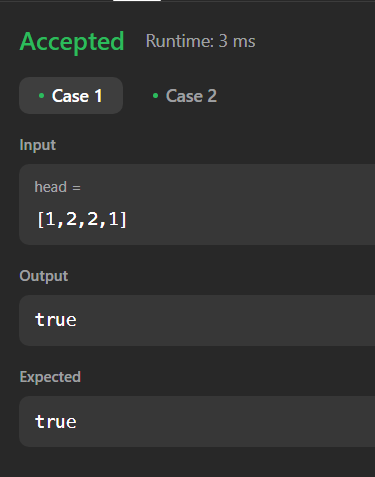
return true;



}

};

# Output:



**Que-3:** [**Middle of the Linked List**](https://leetcode.com/problems/middle-of-the-linked-list/description/)

# Code:

class Solution { public:

ListNode\* middleNode(ListNode\* head) { ListNode\* fast=head,\*slow=head; while(fast!=nullptr && fast->next!=nullptr){

slow=slow->next; fast=fast->next->next;

}



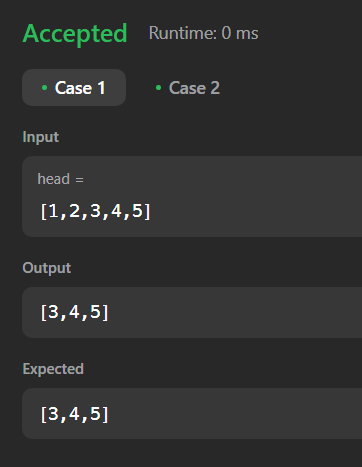


return slow;

}

};

**Output:**



# Que-4: [Add Two Numbers](https://leetcode.com/problems/add-two-numbers/description/)

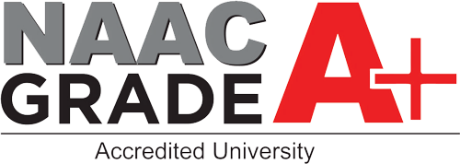
**Code:**

class Solution { public:

ListNode\* addTwoNumbers(ListNode\* l1, ListNode\* l2) { ListNode\* ans=nullptr;

ListNode\* temp=nullptr; int carr=0,val;

while(l1!=nullptr || l2!=nullptr){



if(l1!=nullptr && l2!=nullptr){

val = l1->val + l2->val + carr; l1 = l1->next;

l2 = l2->next;

}

else if(l1 != nullptr) { val = l1->val + carr; l1 = l1->next;

}

else if(l2 != nullptr) { val = l2->val + carr; l2 = l2->next;

}

else {

break;

}

carr=val/10;

if(ans==nullptr){

temp=new ListNode(val%10); ans=temp;

}

else{

temp->next=new ListNode(val%10);





temp = temp->next;

}

}

if(carr!=0){

temp->next=new ListNode(carr);

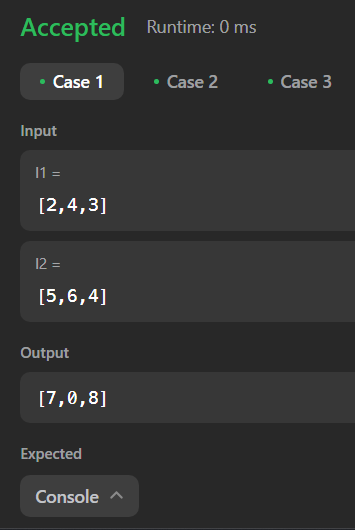
}

return ans;

}

};

# Output:

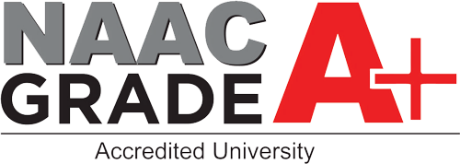


**Que-5:** [**Merge Two Sorted Lists**](https://leetcode.com/problems/merge-two-sorted-lists/description/)

# Code:

class Solution { public:

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



if(list1==nullptr){ return list2;

}

if(list2==nullptr){ return list1;

}

ListNode \*temp=NULL,\*head = NULL; if(list1->val<list2->val){

temp = list1; head=temp;

list1 = list1->next;

}

else{

temp = list2; head=temp;

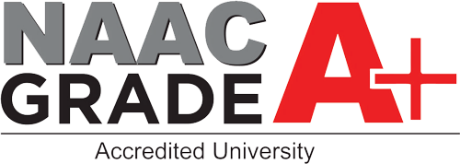
list2 = list2->next;

}

while(list1!=nullptr && list2!=nullptr){ 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 = list1;

}

if(list2!=NULL)

{

temp->next = list2;

}

return head;

}

};

# Output:

