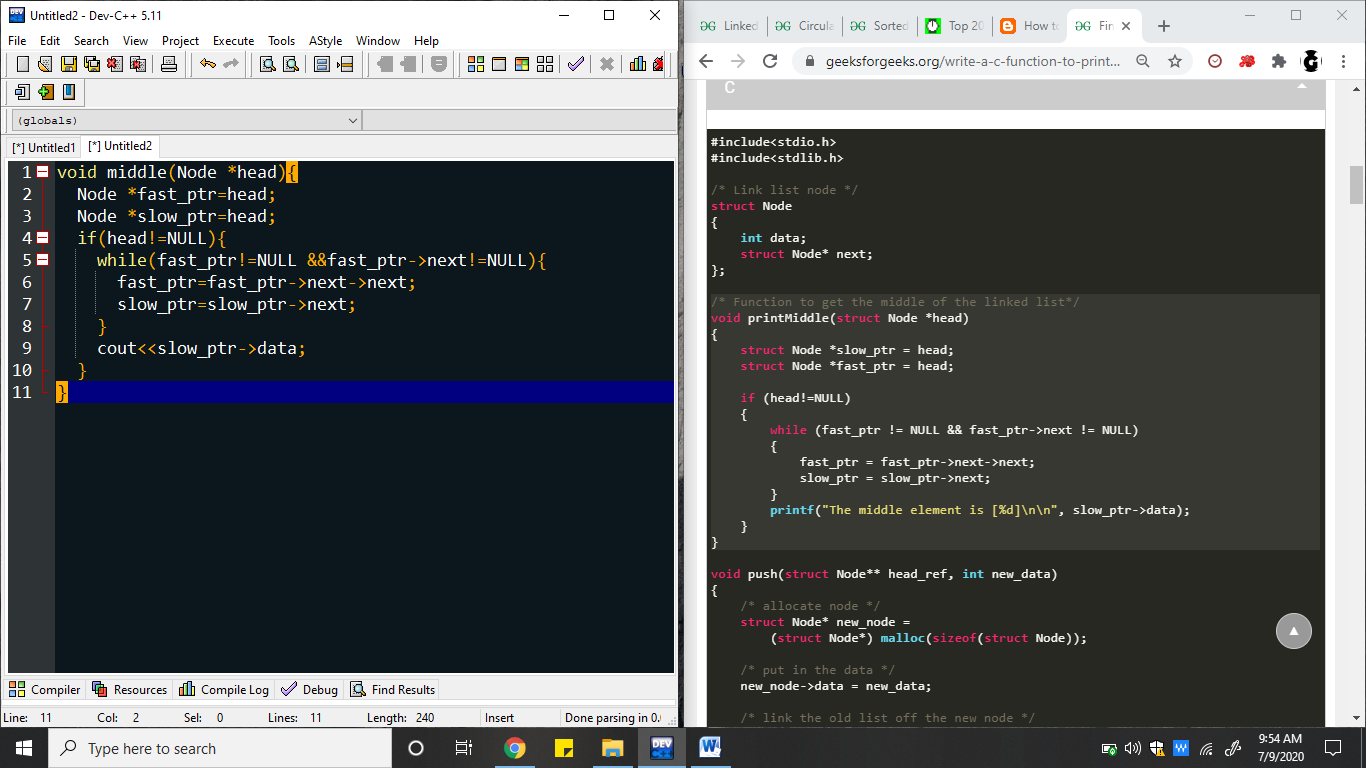
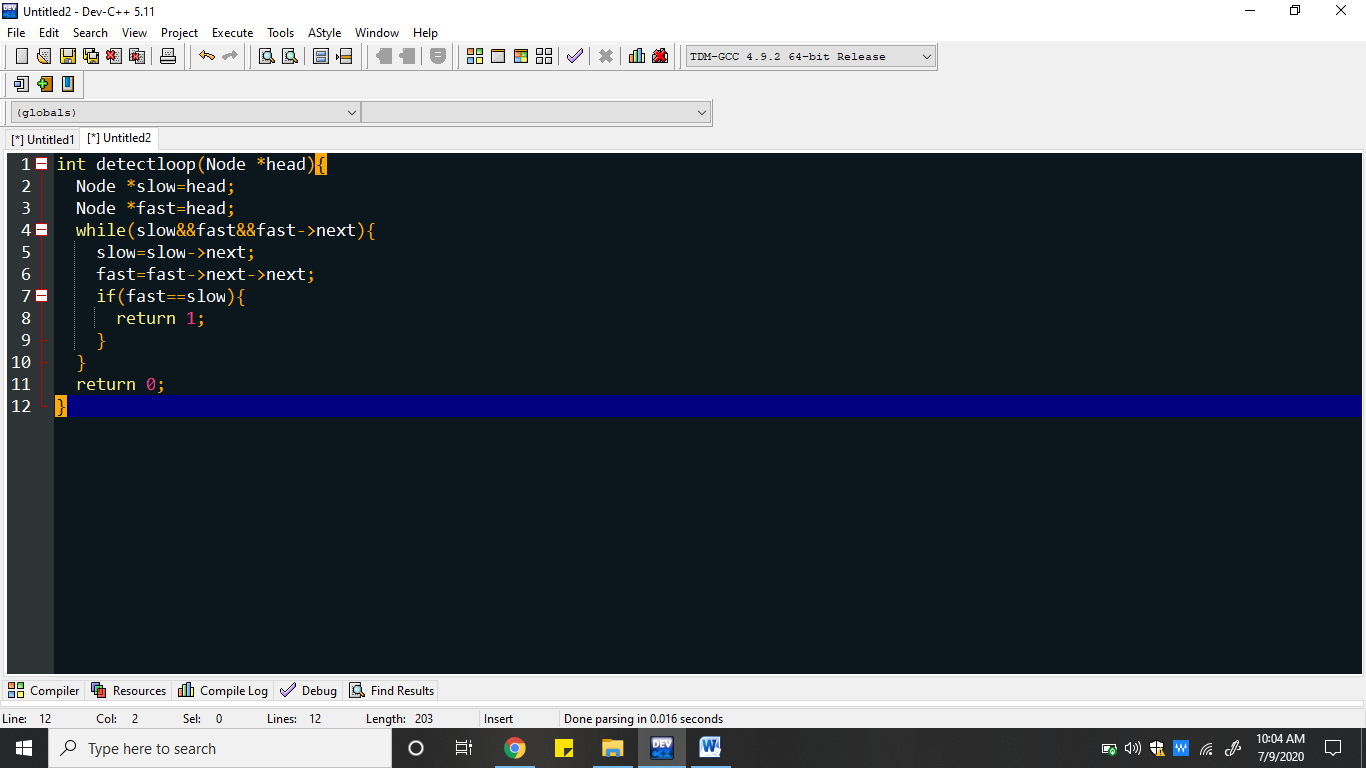
**Q1. How do you find the middle element of a singly linked list in one pass?**

**A1..** 

**Q2**. **How do you check if a given linked list contains a cycle? How do you find the starting node of the cycle?**

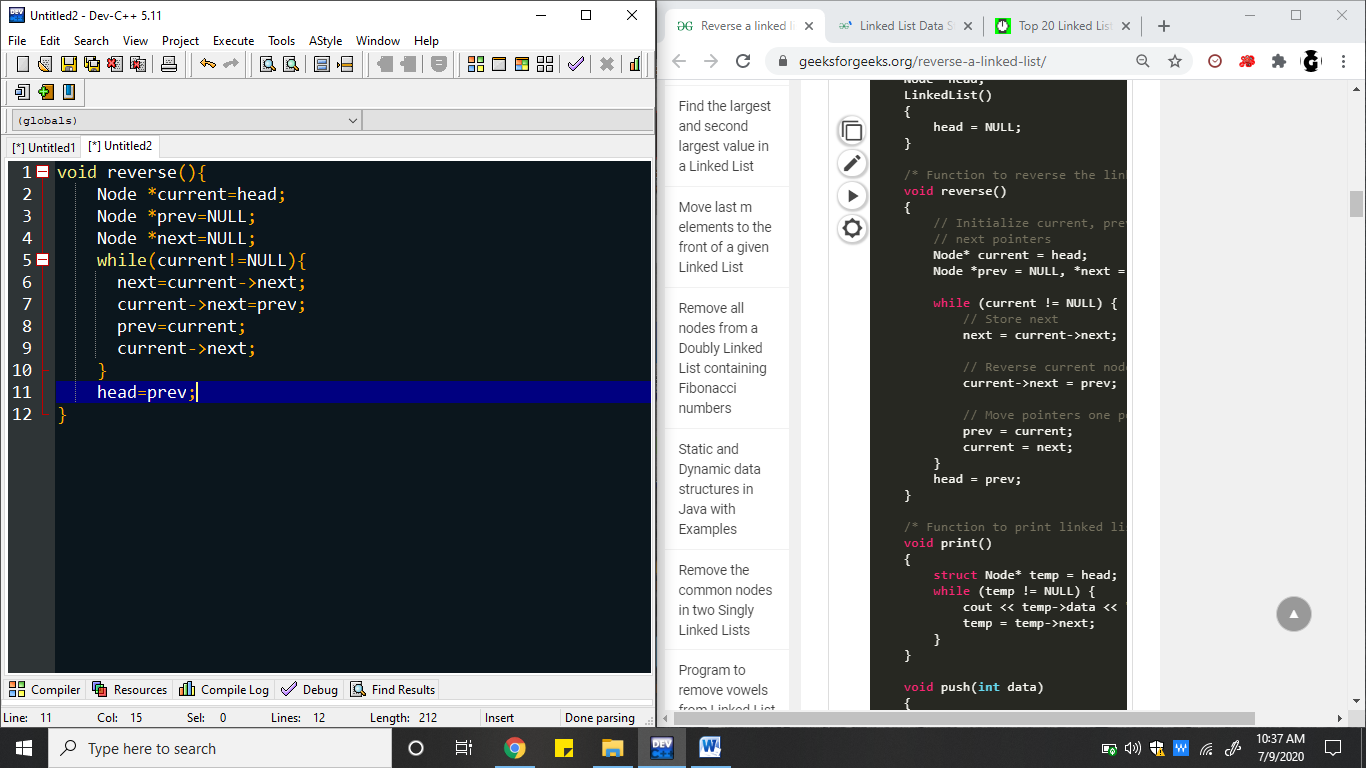
**A2.**



**The point at which fast and slow pointer meets is the first node of the loop.**

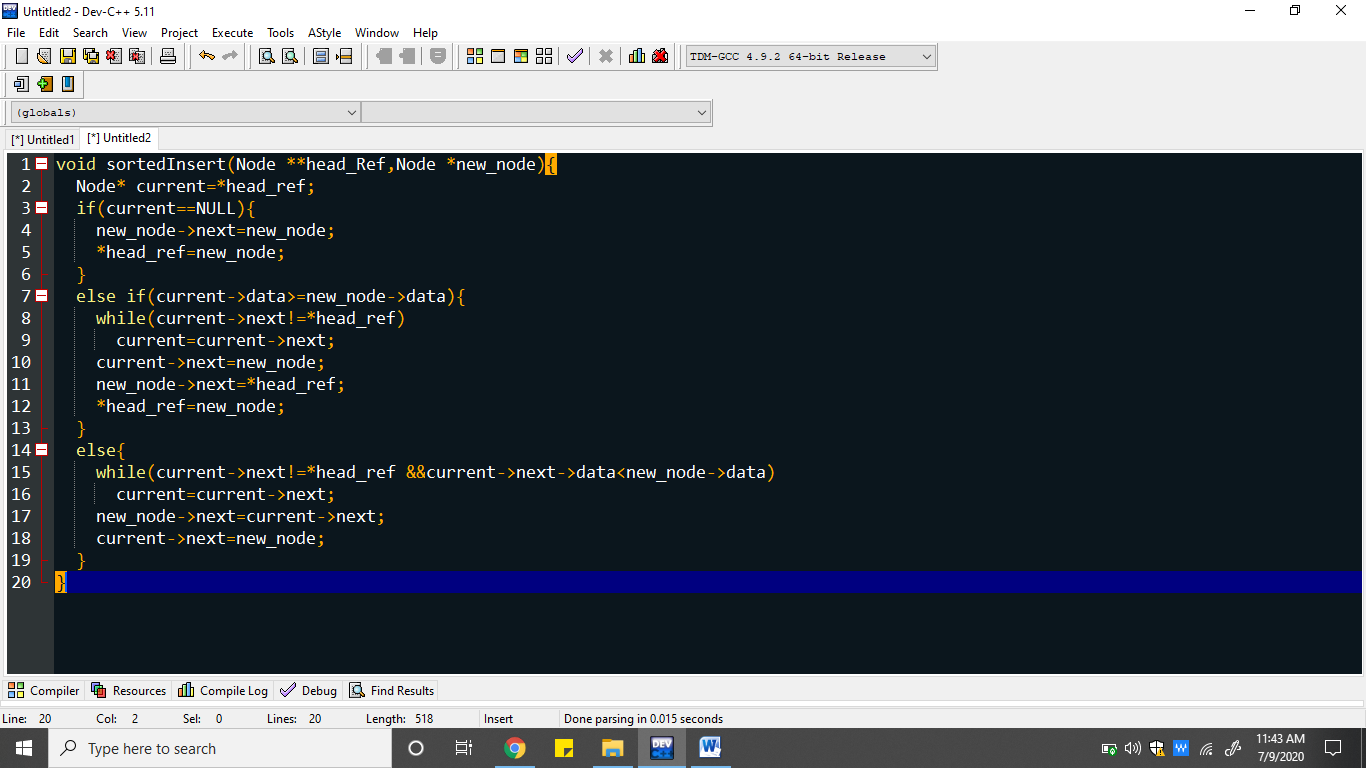
**Q3.** **How do you reverse a linked list?**

**A3.**



**Q4. Sorted insert for circular linked list**

**A4.**

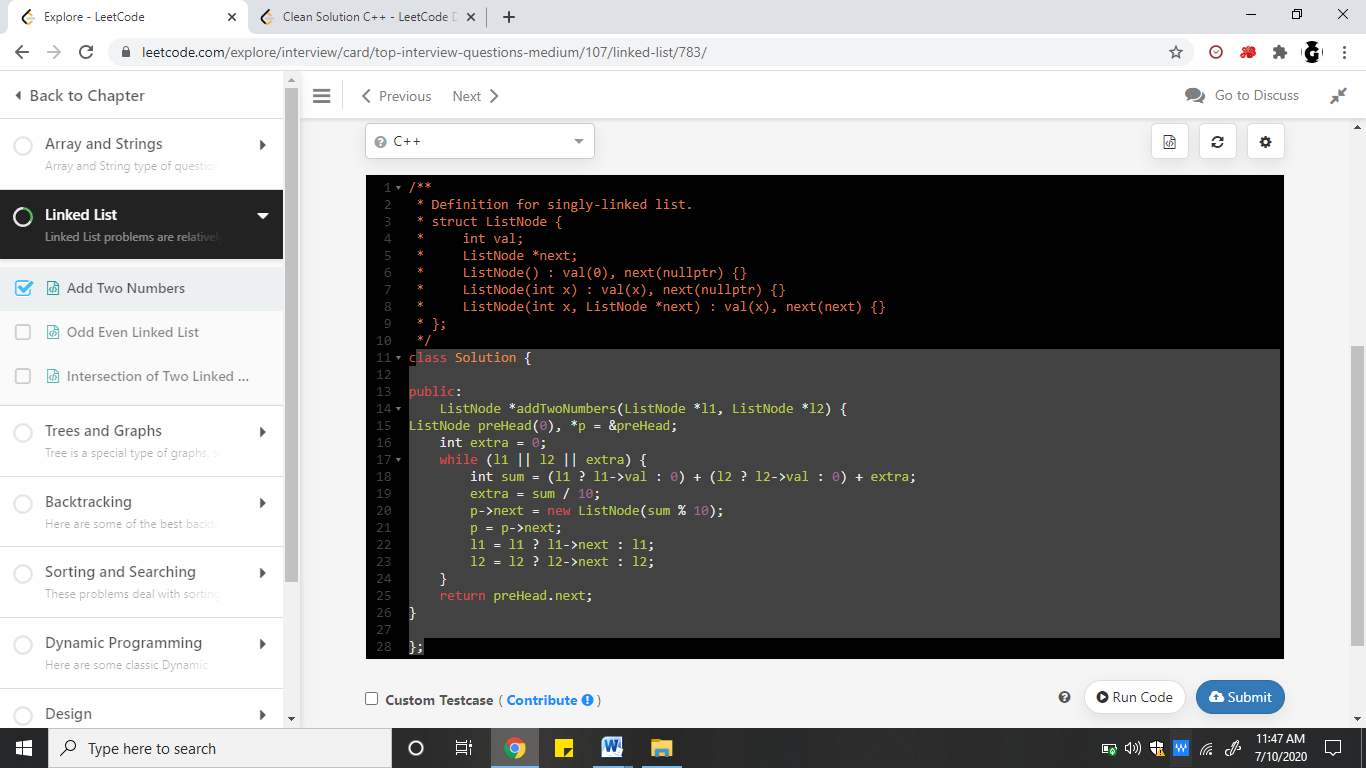


**Q5. You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order and each of their nodes contain a single digit. Add the two numbers and return it as a linked list.**

**You may assume the two numbers do not contain any leading zero, except the number 0 itself.**

**A5. Input:** (2 -> 4 -> 3) + (5 -> 6 -> 4)

**Output:** 7 -> 0 -> 8

**Explanation:** 342 + 465 = 807. 

**===================================================================**

**Q6.Detect loop in linked list**

**bool detectLoop(Node \*head){**

**Node \*slow=head;**

**Node \*fast=head;**

**while(slow&&fast&&fast->next){**

**slow=slow->next;**

**fast=fast->next->next;**

**if(slow==fast){**

**return true**

**}**

**}**

**return false;**

**}**

**}**

**===================================================================**

**Q7.** **Find Intersection Point of Two Linked List**

**Node Count Difference Solution :** Problem can be solved following these steps -

1. Get count of the nodes in the first list, let count be c1.
2. Get a count of the nodes in the second list, let count be c2.
3. Get the difference of counts d = abs(c1 – c2).
4. Now traverse the bigger list from the first node till d nodes so that from here onwards both the lists have equal no of nodes.
5. Then we can traverse both the lists in parallel till we come across a common node. (Note that getting a common node is done by comparing the address of the nodes)

**===================================================================**

**Q8.** **Print Linked List elements .You are given the pointer to the head node of a linked list. You have to print all of its elements in order in a single line.**

**int getIntesectionNode( Node\* head1, Node\* head2)**

**{**

**c1 = getCount(head1)**

**c2 = getCount(head2)**

**d // difference**

**if(c1 > c2)**

**d = c1 - c2**

**return utility(d, head1, head2)**

**else :**

**d = c2 - c1**

**return utility(d, head2, head1)**

**}**

**int utility(d, Node\* head1, Node\* head2)   
{  
 Node\* current1 = head1  
 Node\* current2 = head2  
   
 for ( i = 0 to d-1 )  
 {  
 if(current1 == NULL)   
 return -1  
 current1 = current1->next  
 }  
  
 while(current1 != NULL && current2 != NULL)   
 {  
 if(current1 == current2)   
 return current1->data  
 current1= current1->next  
 current2= current2->next  
 }  
 return -1  
}**

**===================================================================**

**Q9.** **Occurence of an integer in a Linked List**

**void display(Node \*head)**

**{**

**while(head!=NULL){**

**cout<<head->data<<" ";**

**head=head->next;**

**}}**

**Given a singly linked list and a key, count number of occurrences of given key in linked list. For example, if given linked list is 1->2->1->2->1->3->1 and given key is 1, then output should be 4.**

**==================================================================**

**Q10.** **Add two numbers represented by linked lists**

**int count(struct node\* head, int search\_for)**

**{**

**if (head == NULL)**

**return 0;**

**if (head->data == search\_for)**

**return 1 + count(head->next, search\_for);**

**return count(head->next, search\_for);**

**}**

**Given two numbers represented by two linked lists of size N and M. The task is to return a sum list. The sum list is a linked list representation of the addition of two input numbers.**

**Similar problem as Q5.**