public class Lab4 {  
/\*  
Homework  
1. develop an implementation of the equals method in the context of the SinglyLinkedList class.  
class SinglyLinkedList:  
طوير تطبيق أسلوب يساوي في سياق فئة SinglyLinkedList.  
قائمة SinglyLinkedList:  
تعريفه و تدوره  
 def \_\_init\_\_(self):  
 self.head = None  
  
 # Other methods...  
  
 def rotate(self):  
 if self.head is None or self.head.next is None:  
 return  
  
 old\_head = self.head  
 self.head = old\_head.next  
  
 current = self.head  
  
 while current.next:  
 current = current.next  
  
 current.next = old\_head  
 old\_head.next = None  
2. Give an algorithm for finding the second-to-last node in a singly linked list in which the last node is indicated by a null next reference.  
قم بإعطاء خوارزمية للعثور على العقدة الثانية إلى الأخيرة في قائمة مرتبطة بشكل فردي حيث تتم الإشارة إلى العقدة الأخيرة بواسطة مرجع تالٍ فارغ.  
ابحث من الثانيه لاخيره واعمل دواره عشان حلقه  
def find\_second\_to\_last\_node(head):  
 if head is None or head.next is None:  
 return None  
  
 . prev = head  
 current = head  
  
 while current.next is not None:  
 . prev = current  
 current = current.next  
  
 return prev  
  
3. Give an implementation of the size( ) method for the SingularlyLinkedList class, assuming that we did not maintain size as an instance variable.  
3. قم بتنفيذ طريقة size() لفئة SingularlyLinkedList، على افتراض أننا لم نحافظ على الحجم كمتغير مثيل.  
تتبع العنصر يشير عقد الاوللى  
class SinglyLinkedList:  
 def \_\_init\_\_(self):  
 self.head = None  
  
 # Other methods...  
  
 def size(self):  
 count = 0  
 current = self.head  
  
 while current:  
 count += 1  
 current = current.next  
  
 return count  
4. Implement a rotate( ) method in the SinglyLinkedList class, which has semantics equal to addLast(removeFirst( )), yet without creating any new node.  
4. قم بتنفيذ طريقة التدوير () في فئة SinglyLinkedList، والتي لها دلالات تساوي addLast(removeFirst()) ولكن دون إنشاء أي عقدة جديدة.  
COUNTوالثانيه تمرير الحلقه الثانيه زياده  
class SinglyLinkedList:  
 def \_\_init\_\_(self):  
 self.head = None  
  
 # Other methods...  
  
 def size(self):  
 count = 0  
 current = self.head  
  
 while current:  
 count += 1  
 current = current.next  
  
 return count  
5. Describe an algorithm for concatenating two singly linked lists L and M, into a single list L′ that contains all the nodes of L followed by all the nodes of M.  
 وصف خوارزمية لربط قائمتين مرتبطتين بشكل فردي L وM، في قائمة واحدة L′ تحتوي على جميع عقد L متبوعة بجميع عقد M.  
 Lارجاع تنقل تشبر  
def concatenate\_lists(L, M):  
 if L is None:  
 return M  
 if M is None:  
 return L  
  
 L\_prime = L  
 current = L\_prime  
  
 while current.next:  
 current = current.next  
  
 current.next = M  
  
 return L\_prime  
  
6. Describe in detail an algorithm for reversing a singly linked list L using only a constant amount of additional space.  
6. صف بالتفصيل خوارزمية لعكس القائمة المرتبطة بشكل فردي L باستخدام مقدار ثابت فقط من المساحة الإضافية.  
تهين للعقد السابقه تنقلتعين عقد سابقه وعقده حاليه  
def reverse\_linked\_list(head):  
 if head is None or head.next is None:  
 return head  
  
 prev = None  
 current = head  
  
 while current is not None:  
 next\_node = current.next  
 current.next = prev  
 prev = current  
 current = next\_node  
  
 return prev  
  
 \*/  
}