public class Lab5 {  
 /\*  
  
Homework  
1. Consider the implementation of CircularlyLinkedList.addFirst, in Code Fragment 3.16. The else body at lines 39 and 40 of that method relies on a locally declared variable, newest. Redesign that clause to avoid use of any local variable.  
فكر في تنفيذ CircularlyLinkedList.addFirst، في جزء التعليمات البرمجية 3.16. يعتمد نص else الموجود في السطرين 39 و40 من هذه الطريقة على متغير مُعلن محليًا، وهو الأحدث. أعد تصميم هذه الجملة لتجنب استخدام أي متغير محلي.  
def addFirst(self, e):  
 newest = self.\_Node(e, None) # إنشاء عقدة جديدة  
 if self.\_size == 0:  
 newest.next = newest # العقدة الوحيدة تشير إلى نفسها  
 self.\_tail = newest # تحديث العقدة الأخيرة  
 else:  
 newest.next = self.\_tail.next # العقدة الجديدة تشير إلى العقدة الأولى  
 self.\_tail.next = newest # تحديث العقدة الأخيرة  
  
 self.\_size += 1 # زيادة حجم القائم  
2. Give an implementation of the size( ) method for the CircularlyLinkedList class, assuming that we did not maintain size as an instance variable.  
. قم بتنفيذ طريقة size() لفئة CircularlyLinkedList، على افتراض أننا لم نحافظ على الحجم كمتغير مثيل.  
اذا كانت قيمه فاضيه نسوي افتراضيه زياده مغير الثانبه  
def size(self):  
 count = 0  
 if self.\_tail is not None:  
 count = 1  
 current = self.\_tail.next  
 while current != self.\_tail:  
 count += 1  
  
 current = current.next  
 return count  
  
  
  
3. Implement the equals( ) method for the CircularlyLinkedList class, assuming that two lists are equal if they have the same sequence of elements, with corresponding elements currently at the front of the list.  
. قم بتنفيذ طريقة يساوي () لفئة CircularlyLinkedList، على افتراض أن القائمتين متساويتان إذا كان لهما نفس تسلسل العناصر، مع وجود العناصر المقابلة حاليًا في مقدمة القائمة.  
يتم التحقق من الحج  
def equals(self, other):  
 if self.\_size != other.\_size: # التحقق من أن القوائم لديها نفس الحجم  
 return False  
 if self.\_tail is None and other.\_tail is None: # التحقق من أن القوائم فارغة  
  
 return True  
 current\_self = self.\_tail.next  
 current\_other = other.\_tail.next  
 while current\_self != self.\_tail.next: # يتم التحقق من تطابق العناصر  
 if current\_self.element != current\_other.element:  
 return False  
 current\_self = current\_self.next  
 current\_other = current\_other.next  
 return True  
4. Suppose you are given two circularly linked lists, L and M. Describe an algorithm for telling if L and M store the same sequence of elements (but perhaps with different starting points).  
خوارزمية لمقارنة تسلسل العناصر في قوائم متجانستين (L و M):  
يمكننا استخدام خوارزمية بسيطة للتحقق مما إذا كانت القوائم L و M تحتوي على نفس تسلسل العناصر بدءًا من نقاط البداية المختلفة.  
 يمكن القيام بذلك عن طريق إجراء مقارنة تسلسلية بين العناصر في القوائم والتأكد من تطابقها. هنا خوارزمية تنفيذ هذا الفكر:  
- قم بتحديد نقطة البداية في القائمة L.  
- قم بتحديد نقطة البداية في القائمة M.  
- قم بتنفيذ عملية المقارنة التسلسلية بين العناصر في القوائم L و M.  
  
- إذا تطابقت جميع العناصر في القائمتين بنفس الترتيب، فإن القوائم تحتوي على نفس التسلسل.  
  
5. طريقة لتقسيم قائمة متجانسة دائرية L إلى قائمتين متجانستين حجم كل منهما يساوي نصف حجم L:  
  
  
  
  
  
5. Given a circularly linked list L containing an even number of nodes, describe how to split L into two circularly linked lists of half the size.  
في حالة وجود قائمة مرتبطة دائريًا L تحتوي على عدد زوجي من العقد، صف كيفية تقسيم L إلى قائمتين مرتبطتين دائريًا بنصف الحجم.  
حلقه يشير اللى ت=نصف يتم للعصر الاخير  
def splitList(self):  
 if self.\_size % 2 != 0:  
 return "قائمة L يجب أن تحتوي على عدد زوجي من العناصر"  
  
 half\_size = self.\_size // 2  
 current = self.\_tail.next  
  
 count = 0  
 new\_list = CircularlyLinkedList()  
 while count < half\_size:  
 new\_list.addLast(current.element)  
 current = current.next  
 count += 1  
  
 self.\_tail.next = current  
 self.\_size -= half\_size  
 new\_list.\_tail.next = current  
 new\_list.\_size = half\_size  
  
  
 return new\_list  
  
  
  
6. Implement the clone( ) method for the CircularlyLinkedList class.  
6. قم بتنفيذ طريقة الاستنساخ () لفئة CircularlyLinkedList.  
تحقق اذا كان Tفارغه اذا كان كذه ارجاع قيمه فارغه قائمهى جديده  
def clone(self):  
 if self.\_tail is None:  
 return CircularlyLinkedList()  
  
 current = self.\_tail.next  
 new\_list = CircularlyLinkedList()  
 while current != self.\_tail.next:  
  
 new\_list.addLast(current.element)  
 current = current.next  
  
 return new\_list  
  
  
 \*/  
  
}