Topics

* Implement Node Class
* Implement CircularlyLinkedList Class
* Implement Basic Methods of CircularlyLinkedList
* isEmpty()
* size()
* first()
* last()
* addFirst()
* addLast()
* removeFirst()
* rotate()

Homework

* 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.

--def addFirst(self, e):

if self.isEmpty():

self.tail = Node(e)

self.tail.next = self.tail

else:

self.tail.next = Node(e, self.tail.next)

----------------------------------------------------------------------------------------

* Give an implementation of the size( ) method for the CircularlyLinkedList class, assuming that we did not maintain size as an instance variable.

def size(self):

if self.isEmpty():

return 0

count = 1

current = self.tail.next

while current != self.tail:

count += 1

current = current.next

return count

----------------------------------------------------------------------------------------

* 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.

def equals(self, other):

if self.size() != other.size():

return False

current1 = self.tail.next

current2 = other.tail.next

for \_ in range(self.size()):

if current1.data != current2.data:

return False

current1 = current1.next

current2 = current2.next

return True

----------------------------------------------------------------------------------------

* 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).

التأكد من أن الحجم متساويٍ.

التحرك بكل العناصر والتحقق من التطابق.

معالجة قضية الترتيب غير الثابت (الدائرة).

خوارزمية:

تأكد من أن طول L و M متساوي.

جرّب جعل عنصر كل قائمة نقطة بداية، ثم قارن.

--------------------------------------------------

* 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.

الفكرة:

يمكنك التنقل على نصف العقد وقطع الحلقة وربطها في مكان آخر لتقسيم القائمتين.

الطريقة:

إيجاد العنصر الذي يمثل نهاية النصف الأول باستخدام عداد.

إنشاء قائمة جديدة وربطها بالنصف الثاني من القائمة الأصلية.

تعديل النهايات لتكون كل قائمة دائرية.

---------------------------------------------------------------------------------------

* Implement the clone( ) method for the CircularlyLinkedList class.

def clone(self):

new\_list = CircularlyLinkedList()

if self.isEmpty():

return new\_list

current = self.tail.next

for \_ in range(self.size()):

new\_list.addLast(current.data)

current = current.next

return new\_list

-------------------------------------------------------------