public class Lab6 {  
 /\*  
  
  
  
Homework  
1. Describe a method for finding the middle node of a doubly linked list with header and trailer sentinels by “link hopping,” and without relying on explicit knowledge of the size of the list. In the case of an even number of nodes, report the node slightly left of center as the “middle.”  
1. وصف طريقة للعثور على العقدة الوسطى لقائمة مرتبطة بشكل مزدوج مع حراس الرأس والمقطورة عن طريق "التنقل بين الارتباطات"، ودون الاعتماد على معرفة واضحة بحجم القائمة  
. في حالة وجود عدد زوجي من العقد، قم بالإبلاغ عن العقدة التي تقع على يسار المركز قليلًا باعتبارها "الوسطى".  
اضافه عنصر جديد اذا كانت فارغه نفسه اذا لا اللي بعده  
public void addFirst(E e) {  
 Node<E> newest = new Node<>(e);  
 if (isEmpty()) {  
 newest.setNext(newest);  
 tail = newest;  
 } else {  
 newest.setNext(tail.getNext());  
 tail.setNext(newest);  
 }  
 size++;  
}  
  
2. Give an implementation of the size( ) method for the DoublyLinkedList class, assuming that we did not maintain size as an instance variable.  
2. قم بتنفيذ طريقة size() لفئة DoublyLinkedList، على افتراض أننا لم نحافظ على الحجم كمتغير مثيل.  
تحقق من فارغه 0 اذا لا رجع1 يشير عنصر اول دواره  
public int size() {  
 if (isEmpty()) {  
 return 0;  
 }  
  
 int count = 1;  
 Node<E> current = tail.getNext();  
  
 while (current != tail) {  
 count++;  
 current = current.getNext();  
 }  
  
 return count;  
  
}  
3. Implement the equals( ) method for the DoublyLinkedList class.  
3. قم بتنفيذ طريقة يساوي () لفئة DoublyLinkedList.  
يتم تحقق اذا كانت نفس O اذا ايوه صح ويقارن لو ا خ ULL غلط تحقق حجم متساوي صح فرغه صح دواره  
@Override  
public boolean equals(Object obj) {  
 if (this == obj) {  
 return true;  
 }  
  
 if (obj == null || getClass() != obj.getClass()) {  
 return false;  
 }  
  
 CircularlyLinkedList<?> otherList = (CircularlyLinkedList<?>) obj;  
  
  
 if (size() != otherList.size()) {  
 return false;  
 }  
  
 if (isEmpty()) {  
 return true; // Both lists are empty  
 }  
  
 Node<E> currentThis = tail.getNext();  
 Node<?> currentOther = otherList.tail.getNext();  
  
  
 while (currentThis != tail) {  
 if (!currentThis.getElement().equals(currentOther.getElement())) {  
 return false;  
 }  
 currentThis = currentThis.getNext();  
 currentOther = currentOther.getNext();  
 }  
  
 // Compare the last elements  
 return currentThis.getElement().equal  
  
s(currentOther.getElement());  
}  
4. Give an algorithm for concatenating two doubly linked lists L and M, with header and trailer sentinel nodes, into a single list L′.  
 أعط خوارزمية لربط قائمتين مرتبطتين بشكل مزدوج L وM، مع العقد الحارسة للرأس والمقطورة، في قائمة واحدة L′.  
 فارغات صح اذا نل غلط اذا حجم مختلف غلط فارغ صح دواره  
function compareCircularLists(L, M):  
 if L.isEmpty() and M.isEmpty():  
 return true  
  
 startingPointL = findStartingPoint(L)  
 startingPointM = findStartingPoint(M)  
  
 if startingPointL is null or startingPointM is null:  
 return false  
  
  
 currentNodeL = startingPointL  
 currentNodeM = startingPointM  
 completedIterationL = false  
 completedIterationM = false  
  
 while not (completedIterationL and completedIterationM):  
 if currentNodeL.getElement() != currentNodeM.getElement():  
 return false  
  
  
 currentNodeL = currentNodeL.getNext()  
 currentNodeM = currentNodeM.getNext()  
  
 if currentNodeL == startingPointL:  
 completedIterationL = true  
  
 if currentNodeM == startingPointM:  
 completedIterationM = true  
  
  
 if completedIterationL != completedIterationM:  
 return false  
  
 return true  
  
  
5. Our implementation of a doubly linked list relies on two sentinel nodes, header and trailer, but a single sentinel node that guards both ends of the list should suffice. Reimplement the DoublyLinkedList class using only one sentinel node.  
عتمد تنفيذنا للقائمة المرتبطة بشكل مزدوج على عقدتين خافرتين، الرأس والمقطورة،  
 ولكن يجب أن تكون عقدة خافرة واحدة تحمي طرفي القائمة كافية. أعد تنفيذ فئة DoublyLinkedList باستخدام عقدة خافرة واحدة فقط.  
  
 يتم التحقق اذا القيمه فارغه او عدد فردي لانه بيرجع نل لايمكن تقسمه  
 م كسر الرابط المتداخل في النصف الأول من القائمة عن طريق تعيين العقدة الأخيرة في هذا النصف لتشير إلى null. وانشاء اثنين  
function splitCircularList(L):  
 if L is empty or L contains an odd number of nodes:  
 return null  
  
 // Step 1: Find the midpoint  
 Node<E> tortoise = L.getHead();  
 Node<E> hare = L.getHead();  
  
 do {  
 hare = hare.getNext();  
 hare = hare.getNext();  
 tortoise = tortoise.getNext();  
  
 } while (hare != L.getHead() && hare.getNext() != L.getHead());  
  
 // Step 2: Break circular link in the first half  
 Node<E> lastNodeFirstHalf = tortoise;  
 lastNodeFirstHalf.setNext(null);  
  
 // Step 3: Create circular link between the two halves  
 Node<E> lastNodeSecondHalf =  
  
L.getHead();  
 while (lastNodeSecondHalf.getNext() != L.getHead()) {  
 lastNodeSecondHalf = lastNodeSecondHalf.getNext();  
 }  
 lastNodeSecondHalf.setNext(L.getHead());  
  
 // Step 4: Return the two halves  
 CircularlyLinkedList<E> firstHalf = new  
  
CircularlyLinkedList<>();  
 firstHalf.setHead(L.getHead());  
  
 CircularlyLinkedList<E> secondHalf = new CircularlyLinkedList<>();  
 secondHalf.setHead(tortoise);  
  
 return new Pair<>(firstHalf, secondHalf);  
  
  
6. Implement a circular version of a doubly linked list, without any sentinels, that supports all the public behaviors of the original as well as two new update methods, rotate( ) and rotateBackward.  
م بتنفيذ نسخة دائرية من القائمة المرتبطة بشكل مزدوج، بدون أي حراس، والتي تدعم جميع السلوكيات العامة للأصل بالإضافة إلى طريقتين جديدتين للتحديث، التدوير () والتدوير للخلف.  
 انشاء قائمه جديدخ تحقق اذا كانت فلارغه اذا فارغه نستخدم جديده انشاء عقدهT مماثله يتم تحديد الحجم وارجاع  
 public CircularlyLinkedList<E> clone() {  
 CircularlyLinkedList<E> newList = new CircularlyLinkedList<>();  
  
 if (isEmpty()) {  
 return newList; // Return an empty list  
 }  
  
 Node<E> current = tail.getNext();  
 Node<E> newTail = new Node<>(current.getElement());  
  
 newList.tail = newTail;  
  
 while (current.getNext() != tail.getNext()) {  
 current = current.getNext();  
 Node<E> newNode = new Node<>(current.getElement());  
 newTail.setNext(newNode);  
 newTail = newNode;  
 }  
  
 newTail.setNext(newList.tail); // Set the last node to point back to the new tail  
  
 newList.size = size;  
  
 return newList;  
}  
  
  
7. Implement the clone( ) method for the DoublyLinkedList class.  
7. قم بتنفيذ طريقة الاستنساخ () لفئة DoublyLinkedList.  
انشاء قيمه فارغه وتحقق فارغه عقده جديدخP سابقهC القادمه حجم ارجاع  
public DoublyLinkedList<E> clone() {  
 DoublyLinkedList<E> newList = new DoublyLinkedList<>();  
  
 if (isEmpty()) {  
 return newList; // Return an empty list  
 }  
  
 Node<E> current = head;  
 Node<E> newNode = new Node<>(current.getElement(), null, null);  
 newList.head = newNode;  
  
  
 Node<E> prevNode = newNode;  
 current = current.getNext();  
  
 while (current != null) {  
 newNode = new Node<>(current.getElement(), prevNode, null);  
 prevNode.setNext(newNode);  
 prevNode = newNode;  
 current = current.getNext();  
 }  
  
 newList.tail = prevNode;  
 newList.size = size;  
  
 return newList;  
}  
  
 \*/  
}