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.

public void addFirst(E element)

{

if (isEmpty()) {

tail = new Node<>(element, null);

tail.setNext(tail);

} else {

tail.setNext(new Node<>(element, tail.getNext()));

}

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

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

public int size() {

if (isEmpty()) {

return 0;

}

int count = 0;

Node<E> current = tail.getNext();

do {

count++;

current = current.getNext();

} while (current != tail.getNext());

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.

public boolean equals(Object o) {

if (o == this) return true;

if (!(o instanceof CircularlyLinkedList)) return false;

CircularlyLinkedList<?> other = (CircularlyLinkedList<?>) o;

if (this.size() != other.size()) return false;

Node<E> current = this.tail.getNext();

Node<?> otherCurrent = other.tail.getNext();

do {

if (!current.getElement().equals(otherCurrent.getElement())) {

return false;

}

current = current.getNext();

otherCurrent = otherCurrent.getNext();

} while (current != this.tail.getNext());

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).

. إذا كان حجم القائمتين مختلفًا، النتيجة هي "ليستا متساويتين"

.L اختر عقدة بداية في القائمة

قم بمحاولة مطابقة عناصر M مع عناصر L مع الدوران عبر العقد (لتغيير نقطة البداية)

إذا وجدت تطابقًا كاملًا أثناء أي دورة، النتيجة "متساويتان"

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

* 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.

1. استخدم عدادًا للوصول إلى منتصف القائمة.

2. افصل النصف الأول عن الثاني:

اربط آخر عقدة في النصف الأول بالأولى في النصف الأول.

افعل نفس الشيء للنصف الثاني.

3. أنشئ قائمتين جديدتين باستخدام النصفين.

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

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

public CircularlyLinkedList<E> clone() {

CircularlyLinkedList<E> clonedList = new CircularlyLinkedList<>();

if (isEmpty()) {

return clonedList;

}

Node<E> current = tail.getNext();

do {

clonedList.addLast(current.getElement());

current = current.getNext();

} while (current != tail.getNext());

return clonedList;

}