## Topics

1. Implement Node Class
2. Implement CircularlyLinkedList Class
3. Implement Basic Methods of CircularlyLinkedList

* isEmpty()
* size()
* first()
* last()
* addFirst()
* addLast()
* removeFirst()
* rotate()

## 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.

الحل

public void addFirst(E element) {

if (isEmpty()) {

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

tail.setNext(tail);

} else {

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

}

size++;

}

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

الحل

public int size() {

if (isEmpty()) {

return 0;

}

int count = 1; // Start with 1 for the first element

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

while (current != tail) {

count++;

current = current.getNext();

}

return count;

}

1. 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.

الحل

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;

}

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

Node<?> currentOther = otherList.tail.getNext();

while (currentThis != tail && currentThis.getElement().equals(currentOther.getElement())) {

currentThis = currentThis.getNext();

currentOther = currentOther.getNext();

}

return currentThis.getElement().equals(currentOther.getElement());

}

1. 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).
2. 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.
3. Implement the clone( ) method for the CircularlyLinkedList class.