CISC 3130 Ari Mermelstein

Homework #3

An overarching critique that I have received from my students over the last few terms is that the homework assignments do not adequately prepare them for exams. I accept this criticism as constructive, and I have therefore created this assignment which makes you implement your own linked lists. I have found that because students do not program them by themselves, they are unprepared to do so on exams, so hopefully this helps.

Okay, with my soliloquy out of the way, here is the assignment:

The actual linked lists built into C++ work almost exactly like the doubly linked lists described in class, except that they use a very cool heuristic. Instead of maintaining head and tail references such that head.previous = null and tail.next = null, they maintain a reference to one special node called "the sentinel node" or "nil."

This node replaces the null references so that a NullPointerException can never be thrown. The sentinel node has the following properties:

sentinel.previous = what we used to call the tail
sentinel.next = what we used to call the head
what we used to call the tail . next = sentinel
what we used to call the head . previous = sentinel

In effect this node replaces null, and maintaining a reference to it replaces having to maintain separate head and tail references.

I am asking you to implement the following operations that Java7 LinkedLists have. To find their headers and specifications, look here: https://docs.oracle.com/javase/7/docs/api/java/util/LinkedList.html . You can make your lists String specific.

1. public DList()
   This creates the empty list

2. public void addFirst(String elem)
   adds elem to the front of the list

3. public void addLast(String elem)
adds elem to the end of the list

4. public String getFirst()

5. public String getLast()

6. public String removeFirst()
removes the front element of the list and returns it

7. public String removeLast()
removes the last element of the list and returns it.

8. public String get(int index)
Returns the value at "position" index. An IndexOutOfBoundsException should be thrown if the index doesn't exist.

9. public String set(int index, String value)
changes the value at "position" index and returns the old value. An IndexOutOfBoundsException should be thrown if the index doesn't exist.

10. public boolean contains(Object obj)
Returns true if obj appears in the list and false otherwise.

11. public int size()

12. public int indexOf(Object obj)
Returns the index of obj if it is in the list and -1 otherwise.

13. public Iterator<String> iterator()
Returns an iterator over the list. This should be a non static inner class and should work like the one that we described in class.

For extra credit #1: Make the list generic

For extra credit #2: Implement the full on LinkedList class to mimic the java one exactly. Your class should implement List<E> and have all the other additional methods that the LinkedList class has.

Here's some code to get you started:

```
public class DList implements Iterable<String> {

	private static class DListNode {
		public String data;
		public DListNode next;
		public DListNode previous;
	}

	private DListNode nil;


	public DList() {
		nil = new DListNode();
		nil.previous = nil;
		nil.next = nil;
		nil.data = null;
	}

	…

	private class DListIterator implements Iterator<String> {

		private DListNode pointer;

		public DListIterator() {
			if(nil.next ==nil)
				pointer = nil;
			else
				pointer = nil.next;
		}
		…

}
```