```java
import static org.junit.Assert.assertEquals;

/**
 * JUnit test fixture for {@code List<String>}'s constructor and kernel methods.
 *
 * @author Put your name here
 *
 */
public abstract class ListTest {

    /**
     * Invokes the appropriate {@code List} constructor for the implementation
     * under test and returns the result.
     *
     * @return the new list
     * @ensures constructorTest = (<>, <>)
     */
    protected abstract List<String> constructorTest();

    /**
     * Invokes the appropriate {@code List} constructor for the reference
     * implementation and returns the result.
     *
     * @return the new list
     * @ensures constructorRef = (<>, <>)
     */
    protected abstract List<String> constructorRef();

    /**
     * Constructs a {@code List<String>} with the entries in {@code args} and
     * length of the left string equal to {@code leftLength}.
     *
     * @param list
     *            the {@code List} to construct
     * @param leftLength
     *            the length of the left string in the constructed {@code List}
     * @param args
     *            the entries for the list
     * @updates list
     * @requires list = (<>, <>) and 0 <= leftLength <= args.length
     * @ensures <pre>
     * list = ([first leftLength entries in args], [remaining entries in args])
     * </pre>
     */
    private void createFromArgsHelper(List<String> list, int leftLength,
            String... args) {
        for (String s : args) {
            list.addRightFront(s);
            list.advance();
        }
        list.moveToStart();
        for (int i = 0; i < leftLength; i++) {
            list.advance();
        }
    }

    /**
     * Creates and returns a {@code List<String>} of the implementation under
     * test type with the given entries.
     *
     * @param leftLength
     *            the length of the left string in the constructed {@code List}
     * @param args
     *            the entries for the list
     * @return the constructed list
     * @requires 0 <= leftLength <= args.length
     * @ensures <pre>
     * createFromArgs =
     *   ([first leftLength entries in args], [remaining entries in args])
     * </pre>
     */
    protected final List<String> createFromArgsTest(int leftLength,
            String... args) {
        assert 0 <= leftLength : "Violation of: 0 <= leftLength";
        assert leftLength <= args.length : "Violation of: leftLength <= args.length";
        List<String> list = this.constructorTest();
        this.createFromArgsHelper(list, leftLength, args);
        return list;
    }

    /**
     * Creates and returns a {@code List<String>} of the reference
     * implementation type with the given entries.
     *
     * @param leftLength
     *            the length of the left string in the constructed {@code List}
     * @param args
     *            the entries for the list
     * @return the constructed list
     * @requires 0 <= leftLength <= args.length
     * @ensures <pre>
     * createFromArgs =
     *   ([first leftLength entries in args], [remaining entries in args])
     * </pre>
     */
    protected final List<String> createFromArgsRef(int leftLength,
            String... args) {
        assert 0 <= leftLength : "Violation of: 0 <= leftLength";
        assert leftLength <= args.length : "Violation of: leftLength <= args.length";
        List<String> list = this.constructorRef();
        this.createFromArgsHelper(list, leftLength, args);
        return list;
    }

    /*
     * Test cases for constructor, addRightFront, removeRightFront, advance,
     * moveToStart, leftLength, and rightLength.
     */

    @Test
    public final void testConstructor() {
        /*
         * Set up variables and call method under test
         */
        List<String> list1 = this.constructorTest();
        List<String> list2 = this.constructorRef();
        /*
         * Assert that values of variables match expectations
```

```java
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testAddRightFrontLeftEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0);
        List<String> list2 = this.createFromArgsRef(0, "red");
        /*
         * Call method under test
         */
        list1.addRightFront("red");
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testAddRightFrontLeftEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "red", "blue");
        List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
        /*
         * Call method under test
         */
        list1.addRightFront("green");
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testAddRightFrontLeftNonEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
                "purple");
        List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
                "purple", "red");
        /*
         * Call method under test
         */
        list1.addRightFront("red");
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testAddRightFrontLeftNonEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
                "purple");
        List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
                "green", "purple");
        /*
         * Call method under test
         */
        list1.addRightFront("green");
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testRemoveRightFrontLeftEmptyRightOne() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "red");
        List<String> list2 = this.createFromArgsRef(0);
        /*
         * Call method under test
         */
        String s = list1.removeRightFront();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals("red", s);
        assertEquals(list2, list1);
    }

    @Test
    public final void testRemoveRightFrontLeftEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
        List<String> list2 = this.createFromArgsRef(0, "red", "blue");
        /*
         * Call method under test
         */
        String s = list1.removeRightFront();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals("green", s);
        assertEquals(list2, list1);
    }

    @Test
    public final void testRemoveRightFrontLeftNonEmptyRightOne() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
                "purple", "red");
        List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
                "purple");
        /*
         * Call method under test
         */
        String s = list1.removeRightFront();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals("red", s);
        assertEquals(list2, list1);
    }

    @Test
    public final void testRemoveRightFrontLeftNonEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
                "green", "purple");
        List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
                "purple");
        /*
         * Call method under test
         */
        String s = list1.removeRightFront();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals("green", s);
        assertEquals(list2, list1);
    }

    @Test
    public final void testAdvanceLeftEmptyRightOne() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "red");
        List<String> list2 = this.createFromArgsRef(1, "red");
        /*
         * Call method under test
         */
        list1.advance();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testAdvanceLeftEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
        List<String> list2 = this.createFromArgsRef(1, "green", "red", "blue");
        /*
```

```
         * Call method under test
         */
        list1.advance();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testAdvanceLeftNonEmptyRightOne() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
                "purple", "red");
        List<String> list2 = this.createFromArgsRef(4, "yellow", "orange",
                "purple", "red");
        /*
         * Call method under test
         */
        list1.advance();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testAdvanceLeftNonEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
                "green", "purple");
        List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
                "green", "purple");
        /*
         * Call method under test
         */
        list1.advance();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToStartLeftEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0);
        List<String> list2 = this.createFromArgsRef(0);
        /*
         * Call method under test
         */
        list1.moveToStart();
        /*
```

```java
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToStartLeftEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
        List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
        /*
         * Call method under test
         */
        list1.moveToStart();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToStartLeftNonEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
                "purple");
        List<String> list2 = this.createFromArgsRef(0, "yellow", "orange",
                "purple");
        /*
         * Call method under test
         */
        list1.moveToStart();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToStartLeftNonEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
                "green", "purple");
        List<String> list2 = this.createFromArgsRef(0, "yellow", "orange",
                "green", "purple");
        list1.moveToStart();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testRightLengthLeftEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0);
        List<String> list2 = this.createFromArgsRef(0);
        /*
         * Call method under test
         */
        int i = list1.rightLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(0, i);
        assertEquals(list2, list1);
    }

    @Test
    public final void testRightLengthLeftEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
        List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
        /*
         * Call method under test
         */
        int i = list1.rightLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(3, i);
        assertEquals(list2, list1);
    }

    @Test
    public final void testRightLengthLeftNonEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
                "purple");
        List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
                "purple");
        /*
         * Call method under test
         */
        int i = list1.rightLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(0, i);
        assertEquals(list2, list1);
    }

    @Test
    public final void testRightLengthLeftNonEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
                "green", "purple");
        List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
                "green", "purple");
        /*
         * Call method under test
         */
        int i = list1.rightLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(2, i);
        assertEquals(list2, list1);
    }

    @Test
    public final void testLeftLengthLeftEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0);
        List<String> list2 = this.createFromArgsRef(0);
        /*
         * Call method under test
         */
        int i = list1.leftLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(0, i);
        assertEquals(list2, list1);
    }

    @Test
    public final void testLeftLengthLeftEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
        List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
        /*
         * Call method under test
         */
        int i = list1.leftLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(0, i);
        assertEquals(list2, list1);
    }

    @Test
    public final void testLeftLengthLeftNonEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
                "purple");
        List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
```

```java
                "purple");
        /*
         * Call method under test
         */
        int i = list1.leftLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(3, i);
        assertEquals(list2, list1);
    }

    @Test
    public final void testLeftLengthLeftNonEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
                "green", "purple");
        List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
                "green", "purple");
        /*
         * Call method under test
         */
        int i = list1.leftLength();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(2, i);
        assertEquals(list2, list1);
    }

    /*
     * Test cases for iterator.
     */

    @Test
    public final void testIteratorEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0);
        List<String> list2 = this.createFromArgsRef(0);
        List<String> list3 = this.createFromArgsRef(0);
        /*
         * Call method under test
         */
        for (String s : list1) {
            list2.addRightFront(s);
        }
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list3, list1);
        assertEquals(list3, list2);
    }

    @Test
    public final void testIteratorOnlyRight() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "red", "blue");
        List<String> list2 = this.createFromArgsRef(0);
        List<String> list3 = this.createFromArgsRef(0, "red", "blue");
        List<String> list4 = this.createFromArgsRef(0, "blue", "red");
        /*
         * Call method under test
         */
        for (String s : list1) {
            list2.addRightFront(s);
        }
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list3, list1);
        assertEquals(list4, list2);
    }

    @Test
    public final void testIteratorOnlyLeft() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "red", "green", "blue");
        List<String> list2 = this.createFromArgsRef(0);
        List<String> list3 = this.createFromArgsRef(3, "red", "green", "blue");
        List<String> list4 = this.createFromArgsRef(0, "blue", "green", "red");
        /*
         * Call method under test
         */
        for (String s : list1) {
            list2.addRightFront(s);
        }
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list3, list1);
        assertEquals(list4, list2);
    }

    @Test
    public final void testIteratorLeftAndRight() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "purple", "red",
                "green", "blue", "yellow");
        List<String> list2 = this.createFromArgsRef(0);
        List<String> list3 = this.createFromArgsRef(2, "purple", "red", "green",
                "blue", "yellow");
        List<String> list4 = this.createFromArgsRef(0, "yellow", "blue",
                "green", "red", "purple");
        /*
         * Call method under test
         */
        for (String s : list1) {
            list2.addRightFront(s);
        }
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list3, list1);
        assertEquals(list4, list2);
    }

    /*
     * Test cases for other methods: moveToFinish
     */

    @Test
    public final void testMoveToFinishLeftEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0);
        List<String> list2 = this.createFromArgsRef(0);
        /*
         * Call method under test
         */
        list1.moveToFinish();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToFinishLeftEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
        List<String> list2 = this.createFromArgsRef(3, "green", "red", "blue");
        /*
         * Call method under test
         */
        list1.moveToFinish();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToFinishLeftNonEmptyRightEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
                "purple");
        List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
                "purple");
        /*
         * Call method under test
         */
        list1.moveToFinish();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToFinishLeftNonEmptyRightNonEmpty() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
                "green", "purple");
        List<String> list2 = this.createFromArgsRef(4, "yellow", "orange",
                "green", "purple");
        /*
         * Call method under test
         */
        list1.moveToFinish();
        /*
         * Assert that values of variables match expectations
         */
        assertEquals(list2, list1);
    }

    @Test
    public final void testMoveToFinishShowBug() {
        /*
         * Set up variables
         */
        List<String> list1 = this.createFromArgsTest(0);
        List<String> list2 = this.createFromArgsRef(0, "red");
        /*
         * Call method under test
         */
        list1.moveToFinish();
        /*
         * Evaluate the correctness of the result
         */
        list1.addRightFront("red");
        assertEquals(list2, list1);
    }

    // TODO - add test cases for retreat

    /*
     * cannot run tests when left is empty (violates convention)
     */

    // test when left has one element and right has none
    @Test
    public final void testRetreatLeftOneRightEmpty() {
        List<String> list1 = this.createFromArgsTest(1, "orange");
        List<String> list2 = this.createFromArgsRef(0, "orange");

        list1.retreat();
        assertEquals(list1, list2);
    }

    // test when left has multiple elements and right has none
    @Test
    public final void testRetreatLeftNonEmptyRightEmpty() {
        List<String> list1 = this.createFromArgsTest(2, "red", "blue");
        List<String> list2 = this.createFromArgsRef(1, "red", "blue");

        list1.retreat();
        assertEquals(list1, list2);
    }

    // test when left has one element and right has one
    @Test
    public final void testRetreatLeftOneRightOne() {
        List<String> list1 = this.createFromArgsTest(1, "orange", "violet");
        List<String> list2 = this.createFromArgsRef(0, "orange", "violet");

        list1.retreat();
        assertEquals(list1, list2);
    }

    // test when left has one element and right has multiple
    @Test
    public final void testRetreatLeftOneRightNonEmpty() {
        List<String> list1 = this.createFromArgsTest(1, "red", "blue", "yellow",
                "pink");
        List<String> list2 = this.createFromArgsRef(0, "red", "blue", "yellow",
                "pink");

        list1.retreat();
        assertEquals(list1, list2);
    }

    // test when left has multiple elements and right has multiple
    @Test
    public final void testRetreatLeftNonEmptyRightNonEmpty() {
        List<String> list1 = this.createFromArgsTest(2, "red", "blue", "yellow",
                "pink");
        List<String> list2 = this.createFromArgsRef(1, "red", "blue", "yellow",
                "pink");

        list1.retreat();
        assertEquals(list1, list2);
    }

}
```