```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.
```

```
64      *
65      * @param leftLength
66      *            the length of the left string in the constructed {@code List}
67      * @param args
68      *            the entries for the list
69      * @return the constructed list
70      * @requires 0 <= leftLength <= args.length
71      * @ensures <pre>
72      * createFromArgs =
73      *   ([first leftLength entries in args], [remaining entries in args])
74      * </pre>
75      */
76     protected final List<String> createFromArgsTest(int leftLength,
77             String... args) {
78         assert 0 <= leftLength : "Violation of: 0 <= leftLength";
79         assert leftLength <= args.length : "Violation of: leftLength <= args.length";
80         List<String> list = this.constructorTest();
81         this.createFromArgsHelper(list, leftLength, args);
82         return list;
83     }
84
85     /**
86      * Creates and returns a {@code List<String>} of the reference
87      * implementation type with the given entries.
88      *
89      * @param leftLength
90      *            the length of the left string in the constructed {@code List}
91      * @param args
92      *            the entries for the list
93      * @return the constructed list
94      * @requires 0 <= leftLength <= args.length
95      * @ensures <pre>
96      * createFromArgs =
97      *   ([first leftLength entries in args], [remaining entries in args])
98      * </pre>
99      */
100    protected final List<String> createFromArgsRef(int leftLength,
101            String... args) {
102        assert 0 <= leftLength : "Violation of: 0 <= leftLength";
103        assert leftLength <= args.length : "Violation of: leftLength <= args.length";
104        List<String> list = this.constructorRef();
105        this.createFromArgsHelper(list, leftLength, args);
106        return list;
107    }
108
109    /*
110     * Test cases for constructor, addRightFront, removeRightFront, advance,
111     * moveToStart, leftLength, and rightLength.
112     */
113
114    @Test
115    public final void testConstructor() {
116        /*
117         * Set up variables and call method under test
118         */
119        List<String> list1 = this.constructorTest();
120        List<String> list2 = this.constructorRef();
121        /*
122         * Assert that values of variables match expectations
```

```
123          */
124         assertEquals(list2, list1);
125     }
126 
127     @Test
128     public final void testAddRightFrontLeftEmptyRightEmpty() {
129         /*
130          * Set up variables
131          */
132         List<String> list1 = this.createFromArgsTest(0);
133         List<String> list2 = this.createFromArgsRef(0, "red");
134         /*
135          * Call method under test
136          */
137         list1.addRightFront("red");
138         /*
139          * Assert that values of variables match expectations
140          */
141         assertEquals(list2, list1);
142     }
143 
144     @Test
145     public final void testAddRightFrontLeftEmptyRightNonEmpty() {
146         /*
147          * Set up variables
148          */
149         List<String> list1 = this.createFromArgsTest(0, "red", "blue");
150         List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
151         /*
152          * Call method under test
153          */
154         list1.addRightFront("green");
155         /*
156          * Assert that values of variables match expectations
157          */
158         assertEquals(list2, list1);
159     }
160 
161     @Test
162     public final void testAddRightFrontLeftNonEmptyRightEmpty() {
163         /*
164          * Set up variables
165          */
166         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
167                 "purple");
168         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
169                 "purple", "red");
170         /*
171          * Call method under test
172          */
173         list1.addRightFront("red");
174         /*
175          * Assert that values of variables match expectations
176          */
177         assertEquals(list2, list1);
178     }
179 
180     @Test
181     public final void testAddRightFrontLeftNonEmptyRightNonEmpty() {
```

```
182         /*
183          * Set up variables
184          */
185         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
186                 "purple");
187         List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
188                 "green", "purple");
189         /*
190          * Call method under test
191          */
192         list1.addRightFront("green");
193         /*
194          * Assert that values of variables match expectations
195          */
196         assertEquals(list2, list1);
197     }
198
199     @Test
200     public final void testRemoveRightFrontLeftEmptyRightOne() {
201         /*
202          * Set up variables
203          */
204         List<String> list1 = this.createFromArgsTest(0, "red");
205         List<String> list2 = this.createFromArgsRef(0);
206         /*
207          * Call method under test
208          */
209         String s = list1.removeRightFront();
210         /*
211          * Assert that values of variables match expectations
212          */
213         assertEquals("red", s);
214         assertEquals(list2, list1);
215     }
216
217     @Test
218     public final void testRemoveRightFrontLeftEmptyRightNonEmpty() {
219         /*
220          * Set up variables
221          */
222         List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
223         List<String> list2 = this.createFromArgsRef(0, "red", "blue");
224         /*
225          * Call method under test
226          */
227         String s = list1.removeRightFront();
228         /*
229          * Assert that values of variables match expectations
230          */
231         assertEquals("green", s);
232         assertEquals(list2, list1);
233     }
234
235     @Test
236     public final void testRemoveRightFrontLeftNonEmptyRightOne() {
237         /*
238          * Set up variables
239          */
240         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
```

```
241                 "purple", "red");
242         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
243                 "purple");
244         /*
245          * Call method under test
246          */
247         String s = list1.removeRightFront();
248         /*
249          * Assert that values of variables match expectations
250          */
251         assertEquals("red", s);
252         assertEquals(list2, list1);
253     }
254
255     @Test
256     public final void testRemoveRightFrontLeftNonEmptyRightNonEmpty() {
257         /*
258          * Set up variables
259          */
260         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
261                 "green", "purple");
262         List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
263                 "purple");
264         /*
265          * Call method under test
266          */
267         String s = list1.removeRightFront();
268         /*
269          * Assert that values of variables match expectations
270          */
271         assertEquals("green", s);
272         assertEquals(list2, list1);
273     }
274
275     @Test
276     public final void testAdvanceLeftEmptyRightOne() {
277         /*
278          * Set up variables
279          */
280         List<String> list1 = this.createFromArgsTest(0, "red");
281         List<String> list2 = this.createFromArgsRef(1, "red");
282         /*
283          * Call method under test
284          */
285         list1.advance();
286         /*
287          * Assert that values of variables match expectations
288          */
289         assertEquals(list2, list1);
290     }
291
292     @Test
293     public final void testAdvanceLeftEmptyRightNonEmpty() {
294         /*
295          * Set up variables
296          */
297         List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
298         List<String> list2 = this.createFromArgsRef(1, "green", "red", "blue");
299         /*
```

```
300          * Call method under test
301          */
302         list1.advance();
303         /*
304          * Assert that values of variables match expectations
305          */
306         assertEquals(list2, list1);
307     }
308
309     @Test
310     public final void testAdvanceLeftNonEmptyRightOne() {
311         /*
312          * Set up variables
313          */
314         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
315                 "purple", "red");
316         List<String> list2 = this.createFromArgsRef(4, "yellow", "orange",
317                 "purple", "red");
318         /*
319          * Call method under test
320          */
321         list1.advance();
322         /*
323          * Assert that values of variables match expectations
324          */
325         assertEquals(list2, list1);
326     }
327
328     @Test
329     public final void testAdvanceLeftNonEmptyRightNonEmpty() {
330         /*
331          * Set up variables
332          */
333         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
334                 "green", "purple");
335         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
336                 "green", "purple");
337         /*
338          * Call method under test
339          */
340         list1.advance();
341         /*
342          * Assert that values of variables match expectations
343          */
344         assertEquals(list2, list1);
345     }
346
347     @Test
348     public final void testMoveToStartLeftEmptyRightEmpty() {
349         /*
350          * Set up variables
351          */
352         List<String> list1 = this.createFromArgsTest(0);
353         List<String> list2 = this.createFromArgsRef(0);
354         /*
355          * Call method under test
356          */
357         list1.moveToStart();
358         /*
```

```
359          * Assert that values of variables match expectations
360          */
361         assertEquals(list2, list1);
362     }
363
364     @Test
365     public final void testMoveToStartLeftEmptyRightNonEmpty() {
366         /*
367          * Set up variables
368          */
369         List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
370         List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
371         /*
372          * Call method under test
373          */
374         list1.moveToStart();
375         /*
376          * Assert that values of variables match expectations
377          */
378         assertEquals(list2, list1);
379     }
380
381     @Test
382     public final void testMoveToStartLeftNonEmptyRightEmpty() {
383         /*
384          * Set up variables
385          */
386         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
387                 "purple");
388         List<String> list2 = this.createFromArgsRef(0, "yellow", "orange",
389                 "purple");
390         /*
391          * Call method under test
392          */
393         list1.moveToStart();
394         /*
395          * Assert that values of variables match expectations
396          */
397         assertEquals(list2, list1);
398     }
399
400     @Test
401     public final void testMoveToStartLeftNonEmptyRightNonEmpty() {
402         /*
403          * Set up variables
404          */
405         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
406                 "green", "purple");
407         List<String> list2 = this.createFromArgsRef(0, "yellow", "orange",
408                 "green", "purple");
409         list1.moveToStart();
410         /*
411          * Assert that values of variables match expectations
412          */
413         assertEquals(list2, list1);
414     }
415
416     @Test
417     public final void testRightLengthLeftEmptyRightEmpty() {
```

```
418         /*
419          * Set up variables
420          */
421         List<String> list1 = this.createFromArgsTest(0);
422         List<String> list2 = this.createFromArgsRef(0);
423         /*
424          * Call method under test
425          */
426         int i = list1.rightLength();
427         /*
428          * Assert that values of variables match expectations
429          */
430         assertEquals(0, i);
431         assertEquals(list2, list1);
432     }
433 
434     @Test
435     public final void testRightLengthLeftEmptyRightNonEmpty() {
436         /*
437          * Set up variables
438          */
439         List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
440         List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
441         /*
442          * Call method under test
443          */
444         int i = list1.rightLength();
445         /*
446          * Assert that values of variables match expectations
447          */
448         assertEquals(3, i);
449         assertEquals(list2, list1);
450     }
451 
452     @Test
453     public final void testRightLengthLeftNonEmptyRightEmpty() {
454         /*
455          * Set up variables
456          */
457         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
458                 "purple");
459         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
460                 "purple");
461         /*
462          * Call method under test
463          */
464         int i = list1.rightLength();
465         /*
466          * Assert that values of variables match expectations
467          */
468         assertEquals(0, i);
469         assertEquals(list2, list1);
470     }
471 
472     @Test
473     public final void testRightLengthLeftNonEmptyRightNonEmpty() {
474         /*
475          * Set up variables
476          */
```

```
477         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
478                 "green", "purple");
479         List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
480                 "green", "purple");
481         /*
482          * Call method under test
483          */
484         int i = list1.rightLength();
485         /*
486          * Assert that values of variables match expectations
487          */
488         assertEquals(2, i);
489         assertEquals(list2, list1);
490     }
491
492     @Test
493     public final void testLeftLengthLeftEmptyRightEmpty() {
494         /*
495          * Set up variables
496          */
497         List<String> list1 = this.createFromArgsTest(0);
498         List<String> list2 = this.createFromArgsRef(0);
499         /*
500          * Call method under test
501          */
502         int i = list1.leftLength();
503         /*
504          * Assert that values of variables match expectations
505          */
506         assertEquals(0, i);
507         assertEquals(list2, list1);
508     }
509
510     @Test
511     public final void testLeftLengthLeftEmptyRightNonEmpty() {
512         /*
513          * Set up variables
514          */
515         List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
516         List<String> list2 = this.createFromArgsRef(0, "green", "red", "blue");
517         /*
518          * Call method under test
519          */
520         int i = list1.leftLength();
521         /*
522          * Assert that values of variables match expectations
523          */
524         assertEquals(0, i);
525         assertEquals(list2, list1);
526     }
527
528     @Test
529     public final void testLeftLengthLeftNonEmptyRightEmpty() {
530         /*
531          * Set up variables
532          */
533         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
534                 "purple");
535         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
```

```
536                 "purple");
537         /*
538          * Call method under test
539          */
540         int i = list1.leftLength();
541         /*
542          * Assert that values of variables match expectations
543          */
544         assertEquals(3, i);
545         assertEquals(list2, list1);
546     }
547
548     @Test
549     public final void testLeftLengthLeftNonEmptyRightNonEmpty() {
550         /*
551          * Set up variables
552          */
553         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
554                 "green", "purple");
555         List<String> list2 = this.createFromArgsRef(2, "yellow", "orange",
556                 "green", "purple");
557         /*
558          * Call method under test
559          */
560         int i = list1.leftLength();
561         /*
562          * Assert that values of variables match expectations
563          */
564         assertEquals(2, i);
565         assertEquals(list2, list1);
566     }
567
568     /*
569      * Test cases for iterator.
570      */
571
572     @Test
573     public final void testIteratorEmpty() {
574         /*
575          * Set up variables
576          */
577         List<String> list1 = this.createFromArgsTest(0);
578         List<String> list2 = this.createFromArgsRef(0);
579         List<String> list3 = this.createFromArgsRef(0);
580         /*
581          * Call method under test
582          */
583         for (String s : list1) {
584             list2.addRightFront(s);
585         }
586         /*
587          * Assert that values of variables match expectations
588          */
589         assertEquals(list3, list1);
590         assertEquals(list3, list2);
591     }
592
593     @Test
594     public final void testIteratorOnlyRight() {
```

```
595         /*
596          * Set up variables
597          */
598         List<String> list1 = this.createFromArgsTest(0, "red", "blue");
599         List<String> list2 = this.createFromArgsRef(0);
600         List<String> list3 = this.createFromArgsRef(0, "red", "blue");
601         List<String> list4 = this.createFromArgsRef(0, "blue", "red");
602         /*
603          * Call method under test
604          */
605         for (String s : list1) {
606             list2.addRightFront(s);
607         }
608         /*
609          * Assert that values of variables match expectations
610          */
611         assertEquals(list3, list1);
612         assertEquals(list4, list2);
613     }
614 
615     @Test
616     public final void testIteratorOnlyLeft() {
617         /*
618          * Set up variables
619          */
620         List<String> list1 = this.createFromArgsTest(3, "red", "green", "blue");
621         List<String> list2 = this.createFromArgsRef(0);
622         List<String> list3 = this.createFromArgsRef(3, "red", "green", "blue");
623         List<String> list4 = this.createFromArgsRef(0, "blue", "green", "red");
624         /*
625          * Call method under test
626          */
627         for (String s : list1) {
628             list2.addRightFront(s);
629         }
630         /*
631          * Assert that values of variables match expectations
632          */
633         assertEquals(list3, list1);
634         assertEquals(list4, list2);
635     }
636 
637     @Test
638     public final void testIteratorLeftAndRight() {
639         /*
640          * Set up variables
641          */
642         List<String> list1 = this.createFromArgsTest(2, "purple", "red",
643                 "green", "blue", "yellow");
644         List<String> list2 = this.createFromArgsRef(0);
645         List<String> list3 = this.createFromArgsRef(2, "purple", "red", "green",
646                 "blue", "yellow");
647         List<String> list4 = this.createFromArgsRef(0, "yellow", "blue",
648                 "green", "red", "purple");
649         /*
650          * Call method under test
651          */
652         for (String s : list1) {
653             list2.addRightFront(s);
```

```
654         }
655         /*
656          * Assert that values of variables match expectations
657          */
658         assertEquals(list3, list1);
659         assertEquals(list4, list2);
660     }
661
662     /*
663      * Test cases for other methods: moveToFinish
664      */
665
666     @Test
667     public final void testMoveToFinishLeftEmptyRightEmpty() {
668         /*
669          * Set up variables
670          */
671         List<String> list1 = this.createFromArgsTest(0);
672         List<String> list2 = this.createFromArgsRef(0);
673         /*
674          * Call method under test
675          */
676         list1.moveToFinish();
677         /*
678          * Assert that values of variables match expectations
679          */
680         assertEquals(list2, list1);
681     }
682
683     @Test
684     public final void testMoveToFinishLeftEmptyRightNonEmpty() {
685         /*
686          * Set up variables
687          */
688         List<String> list1 = this.createFromArgsTest(0, "green", "red", "blue");
689         List<String> list2 = this.createFromArgsRef(3, "green", "red", "blue");
690         /*
691          * Call method under test
692          */
693         list1.moveToFinish();
694         /*
695          * Assert that values of variables match expectations
696          */
697         assertEquals(list2, list1);
698     }
699
700     @Test
701     public final void testMoveToFinishLeftNonEmptyRightEmpty() {
702         /*
703          * Set up variables
704          */
705         List<String> list1 = this.createFromArgsTest(3, "yellow", "orange",
706                 "purple");
707         List<String> list2 = this.createFromArgsRef(3, "yellow", "orange",
708                 "purple");
709         /*
710          * Call method under test
711          */
712         list1.moveToFinish();
```

```
713         /*
714          * Assert that values of variables match expectations
715          */
716         assertEquals(list2, list1);
717     }
718
719     @Test
720     public final void testMoveToFinishLeftNonEmptyRightNonEmpty() {
721         /*
722          * Set up variables
723          */
724         List<String> list1 = this.createFromArgsTest(2, "yellow", "orange",
725                 "green", "purple");
726         List<String> list2 = this.createFromArgsRef(4, "yellow", "orange",
727                 "green", "purple");
728         /*
729          * Call method under test
730          */
731         list1.moveToFinish();
732         /*
733          * Assert that values of variables match expectations
734          */
735         assertEquals(list2, list1);
736     }
737
738     @Test
739     public final void testMoveToFinishShowBug() {
740         /*
741          * Set up variables
742          */
743         List<String> list1 = this.createFromArgsTest(0);
744         List<String> list2 = this.createFromArgsRef(0, "red");
745         /*
746          * Call method under test
747          */
748         list1.moveToFinish();
749         /*
750          * Evaluate the correctness of the result
751          */
752         list1.addRightFront("red");
753         assertEquals(list2, list1);
754     }
755
756     // TODO - add test cases for retreat
757
758     /*
759      * cannot run tests when left is empty (violates convention)
760      */
761
762     // test when left has one element and right has none
763     @Test
764     public final void testRetreatLeftOneRightEmpty() {
765         List<String> list1 = this.createFromArgsTest(1, "orange");
766         List<String> list2 = this.createFromArgsRef(0, "orange");
767
768         list1.retreat();
769         assertEquals(list1, list2);
770     }
771
```

```
772     // test when left has multiple elements and right has none
773     @Test
774     public final void testRetreatLeftNonEmptyRightEmpty() {
775         List<String> list1 = this.createFromArgsTest(2, "red", "blue");
776         List<String> list2 = this.createFromArgsRef(1, "red", "blue");
777
778         list1.retreat();
779         assertEquals(list1, list2);
780     }
781
782     // test when left has one element and right has one
783     @Test
784     public final void testRetreatLeftOneRightOne() {
785         List<String> list1 = this.createFromArgsTest(1, "orange", "violet");
786         List<String> list2 = this.createFromArgsRef(0, "orange", "violet");
787
788         list1.retreat();
789         assertEquals(list1, list2);
790     }
791
792     // test when left has one element and right has multiple
793     @Test
794     public final void testRetreatLeftOneRightNonEmpty() {
795         List<String> list1 = this.createFromArgsTest(1, "red", "blue", "yellow",
796                 "pink");
797         List<String> list2 = this.createFromArgsRef(0, "red", "blue", "yellow",
798                 "pink");
799
800         list1.retreat();
801         assertEquals(list1, list2);
802     }
803
804     // test when left has multiple elements and right has multiple
805     @Test
806     public final void testRetreatLeftNonEmptyRightNonEmpty() {
807         List<String> list1 = this.createFromArgsTest(2, "red", "blue", "yellow",
808                 "pink");
809         List<String> list2 = this.createFromArgsRef(1, "red", "blue", "yellow",
810                 "pink");
811
812         list1.retreat();
813         assertEquals(list1, list2);
814     }
815
816 }
817
```