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

import org.junit.Test;

import components.list.List;

/**
 * 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
```

```
39      *            the {@code List} to construct
40      * @param leftLength
41      *            the length of the left string in the
    constructed {@code List}
42      * @param args
43      *            the entries for the list
44      * @updates list
45      * @requires list = (<>, <>) and 0 <= leftLength <=
    args.length
46      * @ensures <pre>
47      * list = ([first leftLength entries in args],
    [remaining entries in args])
48      * </pre>
49      */
50     private void createFromArgsHelper(List<String> list,
    int leftLength,
51             String... args) {
52         for (String s : args) {
53             list.addRightFront(s);
54             list.advance();
55         }
56         list.moveToStart();
57         for (int i = 0; i < leftLength; i++) {
58             list.advance();
59         }
60     }
61
62     /**
63      * Creates and returns a {@code List<String>} of the
    implementation under
64      * test type with the given entries.
65      *
66      * @param leftLength
67      *            the length of the left string in the
    constructed {@code List}
68      * @param args
69      *            the entries for the list
70      * @return the constructed list
71      * @requires 0 <= leftLength <= args.length
72      * @ensures <pre>
73      * createFromArgs =
74      *   ([first leftLength entries in args], [remaining
    entries in args])
```

```
 75      * </pre>
 76      */
 77     protected final List<String> createFromArgsTest(int
    leftLength,
 78             String... args) {
 79         assert 0 <= leftLength : "Violation of: 0 <=
    leftLength";
 80         assert leftLength <= args.length : "Violation
    of: leftLength <= args.length";
 81         List<String> list = this.constructorTest();
 82         this.createFromArgsHelper(list, leftLength,
    args);
 83         return list;
 84     }
 85
 86     /**
 87      * Creates and returns a {@code List<String>} of the
    reference
 88      * implementation type with the given entries.
 89      *
 90      * @param leftLength
 91      *            the length of the left string in the
    constructed {@code List}
 92      * @param args
 93      *            the entries for the list
 94      * @return the constructed list
 95      * @requires 0 <= leftLength <= args.length
 96      * @ensures <pre>
 97      * createFromArgs =
 98      *   ([first leftLength entries in args], [remaining
    entries in args])
 99      * </pre>
100      */
101     protected final List<String> createFromArgsRef(int
    leftLength,
102             String... args) {
103         assert 0 <= leftLength : "Violation of: 0 <=
    leftLength";
104         assert leftLength <= args.length : "Violation
    of: leftLength <= args.length";
105         List<String> list = this.constructorRef();
106         this.createFromArgsHelper(list, leftLength,
    args);
```

```
107         return list;
108     }
109 
110     /*
111      * Test cases for constructor, addRightFront,
    removeRightFront, advance,
112      * moveToStart, leftLength, and rightLength.
113      */
114 
115     @Test
116     public final void testConstructor() {
117         /*
118          * Set up variables and call method under test
119          */
120         List<String> list1 = this.constructorTest();
121         List<String> list2 = this.constructorRef();
122         /*
123          * Assert that values of variables match
    expectations
124          */
125         assertEquals(list2, list1);
126     }
127 
128     @Test
129     public final void
    testAddRightFrontLeftEmptyRightEmpty() {
130         /*
131          * Set up variables
132          */
133         List<String> list1 = this.createFromArgsTest(0);
134         List<String> list2 = this.createFromArgsRef(0,
    "red");
135         /*
136          * Call method under test
137          */
138         list1.addRightFront("red");
139         /*
140          * Assert that values of variables match
    expectations
141          */
142         assertEquals(list2, list1);
143     }
144 
```

```
145     @Test
146     public final void
    testAddRightFrontLeftEmptyRightNonEmpty() {
147         /*
148          * Set up variables
149          */
150         List<String> list1 = this.createFromArgsTest(0,
    "red", "blue");
151         List<String> list2 = this.createFromArgsRef(0,
    "green", "red", "blue");
152         /*
153          * Call method under test
154          */
155         list1.addRightFront("green");
156         /*
157          * Assert that values of variables match
    expectations
158          */
159         assertEquals(list2, list1);
160     }
161
162     @Test
163     public final void
    testAddRightFrontLeftNonEmptyRightEmpty() {
164         /*
165          * Set up variables
166          */
167         List<String> list1 = this.createFromArgsTest(3,
    "yellow", "orange",
168                 "purple");
169         List<String> list2 = this.createFromArgsRef(3,
    "yellow", "orange",
170                 "purple", "red");
171         /*
172          * Call method under test
173          */
174         list1.addRightFront("red");
175         /*
176          * Assert that values of variables match
    expectations
177          */
178         assertEquals(list2, list1);
179     }
```

```
180
181     @Test
182     public final void
    testAddRightFrontLeftNonEmptyRightNonEmpty() {
183         /*
184          * Set up variables
185          */
186         List<String> list1 = this.createFromArgsTest(2,
    "yellow", "orange",
187                 "purple");
188         List<String> list2 = this.createFromArgsRef(2,
    "yellow", "orange",
189                 "green", "purple");
190         /*
191          * Call method under test
192          */
193         list1.addRightFront("green");
194         /*
195          * Assert that values of variables match
    expectations
196          */
197         assertEquals(list2, list1);
198     }
199
200     @Test
201     public final void
    testRemoveRightFrontLeftEmptyRightOne() {
202         /*
203          * Set up variables
204          */
205         List<String> list1 = this.createFromArgsTest(0,
    "red");
206         List<String> list2 = this.createFromArgsRef(0);
207         /*
208          * Call method under test
209          */
210         String s = list1.removeRightFront();
211         /*
212          * Assert that values of variables match
    expectations
213          */
214         assertEquals("red", s);
215         assertEquals(list2, list1);
```

```
216     }
217
218     @Test
219     public final void
    testRemoveRightFrontLeftEmptyRightNonEmpty() {
220         /*
221          * Set up variables
222          */
223         List<String> list1 = this.createFromArgsTest(0,
    "green", "red", "blue");
224         List<String> list2 = this.createFromArgsRef(0,
    "red", "blue");
225         /*
226          * Call method under test
227          */
228         String s = list1.removeRightFront();
229         /*
230          * Assert that values of variables match
    expectations
231          */
232         assertEquals("green", s);
233         assertEquals(list2, list1);
234     }
235
236     @Test
237     public final void
    testRemoveRightFrontLeftNonEmptyRightOne() {
238         /*
239          * Set up variables
240          */
241         List<String> list1 = this.createFromArgsTest(3,
    "yellow", "orange",
242                 "purple", "red");
243         List<String> list2 = this.createFromArgsRef(3,
    "yellow", "orange",
244                 "purple");
245         /*
246          * Call method under test
247          */
248         String s = list1.removeRightFront();
249         /*
250          * Assert that values of variables match
    expectations
```

```
251          */
252         assertEquals("red", s);
253         assertEquals(list2, list1);
254     }
255
256     @Test
257     public final void
    testRemoveRightFrontLeftNonEmptyRightNonEmpty() {
258         /*
259          * Set up variables
260          */
261         List<String> list1 = this.createFromArgsTest(2,
    "yellow", "orange",
262                 "green", "purple");
263         List<String> list2 = this.createFromArgsRef(2,
    "yellow", "orange",
264                 "purple");
265         /*
266          * Call method under test
267          */
268         String s = list1.removeRightFront();
269         /*
270          * Assert that values of variables match
    expectations
271          */
272         assertEquals("green", s);
273         assertEquals(list2, list1);
274     }
275
276     @Test
277     public final void testAdvanceLeftEmptyRightOne() {
278         /*
279          * Set up variables
280          */
281         List<String> list1 = this.createFromArgsTest(0,
    "red");
282         List<String> list2 = this.createFromArgsRef(1,
    "red");
283         /*
284          * Call method under test
285          */
286         list1.advance();
287         /*
```

```
288          * Assert that values of variables match
    expectations
289          */
290         assertEquals(list2, list1);
291     }
292
293     @Test
294     public final void
    testAdvanceLeftEmptyRightNonEmpty() {
295         /*
296          * Set up variables
297          */
298         List<String> list1 = this.createFromArgsTest(0,
    "green", "red", "blue");
299         List<String> list2 = this.createFromArgsRef(1,
    "green", "red", "blue");
300         /*
301          * Call method under test
302          */
303         list1.advance();
304         /*
305          * Assert that values of variables match
    expectations
306          */
307         assertEquals(list2, list1);
308     }
309
310     @Test
311     public final void testAdvanceLeftNonEmptyRightOne()
    {
312         /*
313          * Set up variables
314          */
315         List<String> list1 = this.createFromArgsTest(3,
    "yellow", "orange",
316                 "purple", "red");
317         List<String> list2 = this.createFromArgsRef(4,
    "yellow", "orange",
318                 "purple", "red");
319         /*
320          * Call method under test
321          */
322         list1.advance();
```

```
323         /*
324          * Assert that values of variables match
    expectations
325          */
326         assertEquals(list2, list1);
327     }
328 
329     @Test
330     public final void
    testAdvanceLeftNonEmptyRightNonEmpty() {
331         /*
332          * Set up variables
333          */
334         List<String> list1 = this.createFromArgsTest(2,
    "yellow", "orange",
335                 "green", "purple");
336         List<String> list2 = this.createFromArgsRef(3,
    "yellow", "orange",
337                 "green", "purple");
338         /*
339          * Call method under test
340          */
341         list1.advance();
342         /*
343          * Assert that values of variables match
    expectations
344          */
345         assertEquals(list2, list1);
346     }
347 
348     @Test
349     public final void
    testMoveToStartLeftEmptyRightEmpty() {
350         /*
351          * Set up variables
352          */
353         List<String> list1 = this.createFromArgsTest(0);
354         List<String> list2 = this.createFromArgsRef(0);
355         /*
356          * Call method under test
357          */
358         list1.moveToStart();
359         /*
```

```
360          * Assert that values of variables match
    expectations
361          */
362         assertEquals(list2, list1);
363     }
364
365     @Test
366     public final void
    testMoveToStartLeftEmptyRightNonEmpty() {
367         /*
368          * Set up variables
369          */
370         List<String> list1 = this.createFromArgsTest(0,
    "green", "red", "blue");
371         List<String> list2 = this.createFromArgsRef(0,
    "green", "red", "blue");
372         /*
373          * Call method under test
374          */
375         list1.moveToStart();
376         /*
377          * Assert that values of variables match
    expectations
378          */
379         assertEquals(list2, list1);
380     }
381
382     @Test
383     public final void
    testMoveToStartLeftNonEmptyRightEmpty() {
384         /*
385          * Set up variables
386          */
387         List<String> list1 = this.createFromArgsTest(3,
    "yellow", "orange",
388                 "purple");
389         List<String> list2 = this.createFromArgsRef(0,
    "yellow", "orange",
390                 "purple");
391         /*
392          * Call method under test
393          */
394         list1.moveToStart();
```

```
395         /*
396          * Assert that values of variables match
    expectations
397          */
398         assertEquals(list2, list1);
399     }
400
401     @Test
402     public final void
    testMoveToStartLeftNonEmptyRightNonEmpty() {
403         /*
404          * Set up variables
405          */
406         List<String> list1 = this.createFromArgsTest(2,
    "yellow", "orange",
407                 "green", "purple");
408         List<String> list2 = this.createFromArgsRef(0,
    "yellow", "orange",
409                 "green", "purple");
410         list1.moveToStart();
411         /*
412          * Assert that values of variables match
    expectations
413          */
414         assertEquals(list2, list1);
415     }
416
417     @Test
418     public final void
    testRightLengthLeftEmptyRightEmpty() {
419         /*
420          * Set up variables
421          */
422         List<String> list1 = this.createFromArgsTest(0);
423         List<String> list2 = this.createFromArgsRef(0);
424         /*
425          * Call method under test
426          */
427         int i = list1.rightLength();
428         /*
429          * Assert that values of variables match
    expectations
430          */
```

```
431         assertEquals(0, i);
432         assertEquals(list2, list1);
433     }
434 
435     @Test
436     public final void
    testRightLengthLeftEmptyRightNonEmpty() {
437         /*
438          * Set up variables
439          */
440         List<String> list1 = this.createFromArgsTest(0,
    "green", "red", "blue");
441         List<String> list2 = this.createFromArgsRef(0,
    "green", "red", "blue");
442         /*
443          * Call method under test
444          */
445         int i = list1.rightLength();
446         /*
447          * Assert that values of variables match
    expectations
448          */
449         assertEquals(3, i);
450         assertEquals(list2, list1);
451     }
452 
453     @Test
454     public final void
    testRightLengthLeftNonEmptyRightEmpty() {
455         /*
456          * Set up variables
457          */
458         List<String> list1 = this.createFromArgsTest(3,
    "yellow", "orange",
459                 "purple");
460         List<String> list2 = this.createFromArgsRef(3,
    "yellow", "orange",
461                 "purple");
462         /*
463          * Call method under test
464          */
465         int i = list1.rightLength();
466         /*
```

```
467          * Assert that values of variables match
    expectations
468          */
469         assertEquals(0, i);
470         assertEquals(list2, list1);
471     }
472
473     @Test
474     public final void
    testRightLengthLeftNonEmptyRightNonEmpty() {
475         /*
476          * Set up variables
477          */
478         List<String> list1 = this.createFromArgsTest(2,
    "yellow", "orange",
479                 "green", "purple");
480         List<String> list2 = this.createFromArgsRef(2,
    "yellow", "orange",
481                 "green", "purple");
482         /*
483          * Call method under test
484          */
485         int i = list1.rightLength();
486         /*
487          * Assert that values of variables match
    expectations
488          */
489         assertEquals(2, i);
490         assertEquals(list2, list1);
491     }
492
493     @Test
494     public final void
    testLeftLengthLeftEmptyRightEmpty() {
495         /*
496          * Set up variables
497          */
498         List<String> list1 = this.createFromArgsTest(0);
499         List<String> list2 = this.createFromArgsRef(0);
500         /*
501          * Call method under test
502          */
503         int i = list1.leftLength();
```

```
504         /*
505          * Assert that values of variables match
    expectations
506          */
507         assertEquals(0, i);
508         assertEquals(list2, list1);
509     }
510 
511     @Test
512     public final void
    testLeftLengthLeftEmptyRightNonEmpty() {
513         /*
514          * Set up variables
515          */
516         List<String> list1 = this.createFromArgsTest(0,
    "green", "red", "blue");
517         List<String> list2 = this.createFromArgsRef(0,
    "green", "red", "blue");
518         /*
519          * Call method under test
520          */
521         int i = list1.leftLength();
522         /*
523          * Assert that values of variables match
    expectations
524          */
525         assertEquals(0, i);
526         assertEquals(list2, list1);
527     }
528 
529     @Test
530     public final void
    testLeftLengthLeftNonEmptyRightEmpty() {
531         /*
532          * Set up variables
533          */
534         List<String> list1 = this.createFromArgsTest(3,
    "yellow", "orange",
535                 "purple");
536         List<String> list2 = this.createFromArgsRef(3,
    "yellow", "orange",
537                 "purple");
538         /*
```

```
539          * Call method under test
540          */
541         int i = list1.leftLength();
542         /*
543          * Assert that values of variables match
    expectations
544          */
545         assertEquals(3, i);
546         assertEquals(list2, list1);
547     }
548
549     @Test
550     public final void
    testLeftLengthLeftNonEmptyRightNonEmpty() {
551         /*
552          * Set up variables
553          */
554         List<String> list1 = this.createFromArgsTest(2,
    "yellow", "orange",
555                 "green", "purple");
556         List<String> list2 = this.createFromArgsRef(2,
    "yellow", "orange",
557                 "green", "purple");
558         /*
559          * Call method under test
560          */
561         int i = list1.leftLength();
562         /*
563          * Assert that values of variables match
    expectations
564          */
565         assertEquals(2, i);
566         assertEquals(list2, list1);
567     }
568
569     /*
570      * Test cases for iterator.
571      */
572
573     @Test
574     public final void testIteratorEmpty() {
575         /*
576          * Set up variables
```

```
577          */
578         List<String> list1 = this.createFromArgsTest(0);
579         List<String> list2 = this.createFromArgsRef(0);
580         List<String> list3 = this.createFromArgsRef(0);
581         /*
582          * Call method under test
583          */
584         for (String s : list1) {
585             list2.addRightFront(s);
586         }
587         /*
588          * Assert that values of variables match
    expectations
589          */
590         assertEquals(list3, list1);
591         assertEquals(list3, list2);
592     }
593 
594     @Test
595     public final void testIteratorOnlyRight() {
596         /*
597          * Set up variables
598          */
599         List<String> list1 = this.createFromArgsTest(0,
    "red", "blue");
600         List<String> list2 = this.createFromArgsRef(0);
601         List<String> list3 = this.createFromArgsRef(0,
    "red", "blue");
602         List<String> list4 = this.createFromArgsRef(0,
    "blue", "red");
603         /*
604          * Call method under test
605          */
606         for (String s : list1) {
607             list2.addRightFront(s);
608         }
609         /*
610          * Assert that values of variables match
    expectations
611          */
612         assertEquals(list3, list1);
613         assertEquals(list4, list2);
614     }
```

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

```
651          * Call method under test
652          */
653         for (String s : list1) {
654             list2.addRightFront(s);
655         }
656         /*
657          * Assert that values of variables match
    expectations
658          */
659         assertEquals(list3, list1);
660         assertEquals(list4, list2);
661     }
662 
663     /*
664      * Test cases for other methods: moveToFinish
665      */
666 
667     @Test
668     public final void
    testMoveToFinishLeftEmptyRightEmpty() {
669         /*
670          * Set up variables
671          */
672         List<String> list1 = this.createFromArgsTest(0);
673         List<String> list2 = this.createFromArgsRef(0);
674         /*
675          * Call method under test
676          */
677         list1.moveToFinish();
678         /*
679          * Assert that values of variables match
    expectations
680          */
681         assertEquals(list2, list1);
682     }
683 
684     @Test
685     public final void
    testMoveToFinishLeftEmptyRightNonEmpty() {
686         /*
687          * Set up variables
688          */
689         List<String> list1 = this.createFromArgsTest(0,
```

```
    "green", "red", "blue");
690         List<String> list2 = this.createFromArgsRef(3,
    "green", "red", "blue");
691         /*
692          * Call method under test
693          */
694         list1.moveToFinish();
695         /*
696          * Assert that values of variables match
    expectations
697          */
698         assertEquals(list2, list1);
699     }
700
701     @Test
702     public final void
    testMoveToFinishLeftNonEmptyRightEmpty() {
703         /*
704          * Set up variables
705          */
706         List<String> list1 = this.createFromArgsTest(3,
    "yellow", "orange",
707                 "purple");
708         List<String> list2 = this.createFromArgsRef(3,
    "yellow", "orange",
709                 "purple");
710         /*
711          * Call method under test
712          */
713         list1.moveToFinish();
714         /*
715          * Assert that values of variables match
    expectations
716          */
717         assertEquals(list2, list1);
718     }
719
720     @Test
721     public final void
    testMoveToFinishLeftNonEmptyRightNonEmpty() {
722         /*
723          * Set up variables
724          */
```

```
725         List<String> list1 = this.createFromArgsTest(2,
    "yellow", "orange",
726                 "green", "purple");
727         List<String> list2 = this.createFromArgsRef(4,
    "yellow", "orange",
728                 "green", "purple");
729         /*
730          * Call method under test
731          */
732         list1.moveToFinish();
733         /*
734          * Assert that values of variables match
    expectations
735          */
736         assertEquals(list2, list1);
737     }
738
739     @Test
740     public final void testMoveToFinishShowBug() {
741         /*
742          * Set up variables
743          */
744         List<String> list1 = this.createFromArgsTest(0);
745         List<String> list2 = this.createFromArgsRef(0,
    "red");
746         /*
747          * Call method under test
748          */
749         list1.moveToFinish();
750         /*
751          * Evaluate the correctness of the result
752          */
753         list1.addRightFront("red");
754         assertEquals(list2, list1);
755     }
756
757     // TODO - add test cases for retreat
758
759     @Test
760     public final void testRetreatLeft1Right0() {
761         List<String> list1 = this.createFromArgsTest(1,
    "gulugulu");
762         List<String> list2 = this.createFromArgsRef(0,
```

```
   "gulugulu");
763
764         list1.retreat();
765
766         assertEquals(list2, list1);
767     }
768
769     @Test
770     public final void testRetreatLeft1Right1() {
771         List<String> list1 = this.createFromArgsTest(1,
   "red", "green");
772         List<String> list2 = this.createFromArgsRef(0,
   "red", "green");
773
774         list1.retreat();
775
776         assertEquals(list2, list1);
777     }
778
779     @Test
780     public final void testRetreatLeft1RightMany() {
781         List<String> list1 = this.createFromArgsTest(1,
   "red", "green", "blue",
782                 "gulugulu", "galigali");
783         List<String> list2 = this.createFromArgsRef(0,
   "red", "green", "blue",
784                 "gulugulu", "galigali");
785
786         list1.retreat();
787
788         assertEquals(list2, list1);
789     }
790
791     @Test
792     public final void testRetreatLeftManyRight0() {
793         List<String> list1 = this.createFromArgsTest(5,
   "red", "green", "blue",
794                 "gulugulu", "galigali");
795         List<String> list2 = this.createFromArgsRef(4,
   "red", "green", "blue",
796                 "gulugulu", "galigali");
797
798         list1.retreat();
```

```
799
800         assertEquals(list2, list1);
801     }
802
803     @Test
804     public final void testRetreatLeftManyRight1() {
805         List<String> list1 = this.createFromArgsTest(5,
    "red", "green", "blue",
806                 "gulugulu", "galigali",
    "honglonghonglong");
807         List<String> list2 = this.createFromArgsRef(4,
    "red", "green", "blue",
808                 "gulugulu", "galigali",
    "honglonghonglong");
809
810         list1.retreat();
811
812         assertEquals(list2, list1);
813     }
814
815     @Test
816     public final void testRetreatLeftManyRightMany() {
817         List<String> list1 = this.createFromArgsTest(5,
    "red", "green", "blue",
818                 "gulugulu", "galigali",
    "honglonghonglong", "honglonghonglong");
819         List<String> list2 = this.createFromArgsRef(4,
    "red", "green", "blue",
820                 "gulugulu", "galigali",
    "honglonghonglong", "honglonghonglong");
821
822         list1.retreat();
823
824         assertEquals(list2, list1);
825     }
826
827     @Test
828     public final void testRetreatWithSameElement() {
829         List<String> list1 = this.createFromArgsTest(2,
    "red", "red", "red",
830                 "red");
831         List<String> list2 = this.createFromArgsRef(2,
    "red", "red", "red",
```

```
832                 "red");
833
834         list1.retreat();
835
836         assertNotEquals(list2, list1);
837         // <"red","red"><"red,"red"> is not equal to
    <"red","red","red","red">
838     }
839
840 }
841
```