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

public class GlossaryTest {

    @Test
    public void getMapTest_1() {
        // normal map
        String fileName = "get_map_test_1.txt";
        SimpleWriter wordDefinitionTest = new SimpleWriter1L(fileName);
        wordDefinitionTest.println("amazing");
        wordDefinitionTest.println("my lifestyle");
        wordDefinitionTest.println();
        wordDefinitionTest.println("horrible");
        wordDefinitionTest.println("not my lifestyle");
        wordDefinitionTest.println();
        wordDefinitionTest.println("mid");
        wordDefinitionTest.println("sometimes my lifestyle");
        wordDefinitionTest.println();
        Map<String, String> mapExpected = new Map1L<>();
        Map<String, String> map = new Map1L<>();
        mapExpected.add("amazing", "my lifestyle ");
        mapExpected.add("horrible", "not my lifestyle ");
        mapExpected.add("mid", "sometimes my lifestyle ");

        Glossary.getMap(fileName, map);
        assertEquals(mapExpected, map);

    }

    @Test
    public void getMapTest_2() {
        // empty file
        String fileName = "get_map_test_2.txt";
        SimpleWriter wordDefinitionTest = new SimpleWriter1L(fileName);
        Map<String, String> mapExpected = new Map1L<>();
        Map<String, String> map = new Map1L<>();

        Glossary.getMap(fileName, map);
        assertEquals(mapExpected, map);

    }

```

```
56      @Test
57      public void getMapTest_3() {
58          // definition is more than one line
59          String fileName = "get_map_test_3.txt";
60          SimpleWriter wordDefinitionTest = new
    SimpleWriter1L(fileName);
61          wordDefinitionTest.println("amazing");
62          wordDefinitionTest.println("my lifestyle");
63          wordDefinitionTest.println();
64          wordDefinitionTest.println("horrible");
65          wordDefinitionTest.println("not my lifestyle");
66          wordDefinitionTest.println("but it is everyone
    else's");
67          wordDefinitionTest.println();
68          wordDefinitionTest.println("mid");
69          wordDefinitionTest.println("sometimes my lifestyle");
70          wordDefinitionTest.println();
71          Map<String, String> mapExpected = new Map1L<>();
72          Map<String, String> map = new Map1L<>();
73          mapExpected.add("amazing", "my lifestyle ");
74          mapExpected.add("horrible",
75                  "not my lifestyle but it is everyone else's ");
76          mapExpected.add("mid", "sometimes my lifestyle ");
77
78          Glossary.getMap(fileName, map);
79          assertEquals(mapExpected, map);
80      }
81
82      @Test
83      public void wordQueueTest_1() {
84          // normal word queue
85          String fileName = "word_queue_test_1.txt";
86          SimpleWriter wordDefinitionTest = new
    SimpleWriter1L(fileName);
87          wordDefinitionTest.println("amazing");
88          wordDefinitionTest.println("my lifestyle");
89          wordDefinitionTest.println();
90          wordDefinitionTest.println("horrible");
91          wordDefinitionTest.println("not my lifestyle");
92          wordDefinitionTest.println();
93          wordDefinitionTest.println("mid");
94          wordDefinitionTest.println("sometimes my lifestyle");
95          wordDefinitionTest.println();
96
```

```
 97         Map<String, String> mapExpected = new Map1L<>();
 98         mapExpected.add("amazing", "my lifestyle");
 99         mapExpected.add("horrible", "not my lifestyle");
100         mapExpected.add("mid", "sometimes my lifestyle");
101         Map<String, String> map = new Map1L<>();
102         map.add("amazing", "my lifestyle");
103         map.add("horrible", "not my lifestyle");
104         map.add("mid", "sometimes my lifestyle");
105         Queue<String> wordBank = new Queue1L<>();
106         Queue<String> wordBankExpected = new Queue1L<>();
107         wordBankExpected.enqueue("amazing");
108         wordBankExpected.enqueue("horrible");
109         wordBankExpected.enqueue("mid");
110 
111         Glossary.wordQueue(map, wordBank);
112         assertEquals(wordBankExpected, wordBank);
113         assertEquals(mapExpected, map);
114 
115     }
116 
117     @Test
118     public void wordQueueTest_2() {
119         // words not in order
120         String fileName = "word_queue_test_2.txt";
121         SimpleWriter wordDefinitionTest = new
    SimpleWriter1L(fileName);
122         wordDefinitionTest.println("zebra");
123         wordDefinitionTest.println("my lifestyle");
124         wordDefinitionTest.println();
125         wordDefinitionTest.println("horrible");
126         wordDefinitionTest.println("not my lifestyle");
127         wordDefinitionTest.println();
128         wordDefinitionTest.println("mid");
129         wordDefinitionTest.println("sometimes my lifestyle");
130         wordDefinitionTest.println();
131 
132         Map<String, String> mapExpected = new Map1L<>();
133         mapExpected.add("zebra", "my lifestyle");
134         mapExpected.add("horrible", "not my lifestyle");
135         mapExpected.add("mid", "sometimes my lifestyle");
136         Map<String, String> map = new Map1L<>();
137         map.add("zebra", "my lifestyle");
138         map.add("horrible", "not my lifestyle");
139         map.add("mid", "sometimes my lifestyle");
```

```
140         Queue<String> wordBank = new Queue1L<>();
141         Queue<String> wordBankExpected = new Queue1L<>();
142         wordBankExpected.enqueue("horrible");
143         wordBankExpected.enqueue("mid");
144         wordBankExpected.enqueue("zebra");
145 
146         Glossary.wordQueue(map, wordBank);
147         assertEquals(wordBankExpected, wordBank);
148         assertEquals(mapExpected, map);
149 
150     }
151 
152     @Test
153     public void wordQueueTest_3() {
154         // empty word
155         String fileName = "word_queue_test_3.txt";
156         SimpleWriter wordDefinitionTest = new
    SimpleWriter1L(fileName);
157         Map<String, String> mapExpected = new Map1L<>();
158         Map<String, String> map = new Map1L<>();
159         Queue<String> wordBank = new Queue1L<>();
160         Queue<String> wordBankExpected = new Queue1L<>();
161 
162         Glossary.wordQueue(map, wordBank);
163         assertEquals(wordBankExpected, wordBank);
164         assertEquals(mapExpected, map);
165     }
166 
167     @Test
168     public void generateElementsTest_1() {
169         // regular input
170         String str = "hello";
171         Set<Character> charSet = new Set1L<>();
172         Set<Character> charSetExpected = new Set1L<>();
173         charSetExpected.add('h');
174         charSetExpected.add('e');
175         charSetExpected.add('l');
176         charSetExpected.add('o');
177 
178         Glossary.generateElements(str, charSet);
179         assertEquals(charSetExpected, charSet);
180     }
181 
182     @Test
```

```
183     public void generateElementsTest_2() {
184         // input with space
185         String str = "hello ";
186         Set<Character> charSet = new Set1L<>();
187         Set<Character> charSetExpected = new Set1L<>();
188         charSetExpected.add('h');
189         charSetExpected.add('e');
190         charSetExpected.add('l');
191         charSetExpected.add('o');
192         charSetExpected.add(' ');
193 
194         Glossary.generateElements(str, charSet);
195         assertEquals(charSetExpected, charSet);
196     }
197 
198     @Test
199     public void generateElementsTest_3() {
200         // input with comma
201         String str = "hell,o";
202         Set<Character> charSet = new Set1L<>();
203         Set<Character> charSetExpected = new Set1L<>();
204         charSetExpected.add('h');
205         charSetExpected.add('e');
206         charSetExpected.add('l');
207         charSetExpected.add('o');
208         charSetExpected.add(',');
209 
210         Glossary.generateElements(str, charSet);
211         assertEquals(charSetExpected, charSet);
212     }
213 
214     @Test
215     public void nextWordOrSeparatorTest_1() {
216         // input without separator
217 
218         int position = 0;
219         Set<Character> separators = new Set1L<>();
220         separators.add(' ');
221         String text = "nickcheong ";
222         String resultExpected = "nickcheong";
223         String result = Glossary.nextWordOrSeparator(text,
    position,
224                 separators);
225         assertEquals(resultExpected, result);
```

```
226
227     }
228
229     @Test
230     public void nextWordOrSeparatorTest_2() {
231         // normal input - printing the first word
232         int position = 0;
233         Set<Character> separators = new Set1L<>();
234         separators.add(' ');
235         String text = "nick cheong ";
236         String resultExpected = "nick";
237         String result = Glossary.nextWordOrSeparator(text,
    position,
238                 separators);
239         assertEquals(resultExpected, result);
240
241     }
242
243     @Test
244     // printing the separator
245     public void nextWordOrSeparatorTest_3() {
246         int position = 4;
247         Set<Character> separators = new Set1L<>();
248         separators.add(' ');
249         String text = "nick cheong";
250         String resultExpected = " ";
251         String result = Glossary.nextWordOrSeparator(text,
    position,
252                 separators);
253
254         assertEquals(resultExpected, result);
255
256     }
257
258     @Test
259     // printing the second word
260     public void nextWordOrSeparatorTest_4() {
261         int position = 5;
262         Set<Character> separators = new Set1L<>();
263         separators.add(' ');
264         String text = "nick cheong";
265         String resultExpected = "cheong";
266         String result = Glossary.nextWordOrSeparator(text,
    position,
```

```
267                 separators);
268
269         assertEquals(resultExpected, result);
270
271     }
272
273     @Test
274     public void generateHTMLTest_1() {
275         // no links in definition
276         Map<String, String> wordDefinition = new Map1L<>();
277         wordDefinition.add("nick", "the coolest person");
278         Map<String, String> wordDefinitionExpected = new
    Map1L<>();
279         wordDefinitionExpected.add("nick", "the coolest
    person");
280         Queue<String> wordBank = new Queue1L<>();
281         wordBank.enqueue("nick");
282         Queue<String> wordBankExpected = new Queue1L<>();
283         wordBankExpected.enqueue("nick");
284         SimpleWriter out = new
    SimpleWriter1L("generate_HTML_test_1.txt");
285         SimpleWriter outFile = new SimpleWriter1L(
286                 "generate_HTML_subtest_1.txt");
287
288         SimpleReader in = new
    SimpleReader1L("generate_HTML_test_1.txt");
289
290         String outputFile = "test";
291         String word = "nick";
292         String wordExpected = "nick";
293         String definition = "the coolest person";
294         String definitionExpected = "the coolest person";
295
296         Glossary.generateHTML(wordDefinitionExpected, word,
    definition, out,
297                 outputFile);
298         assertEquals("<a href = nick.html>", in.nextLine());
299         assertEquals("nick", in.nextLine());
300         assertEquals("</a>", in.nextLine());
301         SimpleReader inFile = new SimpleReader1L(
302                 outputFile + "/" + word + ".html");
303         assertEquals("<head>", inFile.nextLine());
304         assertEquals("<title>", inFile.nextLine());
305         assertEquals(wordExpected, inFile.nextLine());
```

```
306         assertEquals("</title>", inFile.nextLine());
307         assertEquals("</head>", inFile.nextLine());
308         assertEquals("<body>", inFile.nextLine());
309         assertEquals("<h2>", inFile.nextLine());
310         assertEquals("<b>", inFile.nextLine());
311         assertEquals("<i>", inFile.nextLine());
312         assertEquals("<font color = \"red\">" + wordExpected +
    "</font>",
313                 inFile.nextLine());
314         assertEquals("</i>", inFile.nextLine());
315         assertEquals("</b>", inFile.nextLine());
316         assertEquals("</h2>", inFile.nextLine());
317         assertEquals("<blockquote>", inFile.nextLine());
318         assertEquals(definitionExpected, inFile.nextLine());
319         assertEquals("</blockquote>", inFile.nextLine());
320         assertEquals("<hr>", inFile.nextLine());
321         assertEquals("<p>", inFile.nextLine());
322         assertEquals("Return to", inFile.nextLine());
323         assertEquals("<a href = index.html>",
    inFile.nextLine());
324         assertEquals("index</a>.", inFile.nextLine());
325         assertEquals("</p>", inFile.nextLine());
326         assertEquals("</body>", inFile.nextLine());
327         assertEquals("</html>", inFile.nextLine());
328     }
329
330     @Test
331     public void generateHTMLTest_2() {
332         // links in definition
333         Map<String, String> wordDefinition = new Map1L<>();
334         wordDefinition.add("nick", "the coolest");
335         wordDefinition.add("coolest", "something awesome");
336         Map<String, String> wordDefinitionExpected = new
    Map1L<>();
337         wordDefinitionExpected.add("nick", "the coolest");
338         wordDefinitionExpected.add("coolest", "something
    awesome");
339         Queue<String> wordBank = new Queue1L<>();
340         wordBank.enqueue("nick");
341         Queue<String> wordBankExpected = new Queue1L<>();
342         wordBankExpected.enqueue("nick");
343         SimpleWriter out = new
    SimpleWriter1L("generate_HTML_test_2.txt");
344         SimpleWriter outFile = new SimpleWriter1L(
```

```
345                 "generate_HTML_subtest_2.txt");
346
347         SimpleReader in = new
    SimpleReader1L("generate_HTML_test_2.txt");
348
349         String outputFile = "test";
350         String word = "nick";
351         String wordExpected = "nick";
352         String definition = "the coolest";
353         String definitionExpected = "the <a href =
    coolest.html>coolest</a>";
354
355         Glossary.generateHTML(wordDefinitionExpected, word,
    definition, out,
356                 outputFile);
357         assertEquals("<a href = nick.html>", in.nextLine());
358         assertEquals("nick", in.nextLine());
359         assertEquals("</a>", in.nextLine());
360         SimpleReader inFile = new SimpleReader1L(
361                 outputFile + "/" + word + ".html");
362         assertEquals("<head>", inFile.nextLine());
363         assertEquals("<title>", inFile.nextLine());
364         assertEquals(wordExpected, inFile.nextLine());
365         assertEquals("</title>", inFile.nextLine());
366         assertEquals("</head>", inFile.nextLine());
367         assertEquals("<body>", inFile.nextLine());
368         assertEquals("<h2>", inFile.nextLine());
369         assertEquals("<b>", inFile.nextLine());
370         assertEquals("<i>", inFile.nextLine());
371         assertEquals("<font color = \"red\">" + wordExpected +
    "</font>",
372                 inFile.nextLine());
373         assertEquals("</i>", inFile.nextLine());
374         assertEquals("</b>", inFile.nextLine());
375         assertEquals("</h2>", inFile.nextLine());
376         assertEquals("<blockquote>", inFile.nextLine());
377         assertEquals(definitionExpected, inFile.nextLine());
378         assertEquals("</blockquote>", inFile.nextLine());
379         assertEquals("<hr>", inFile.nextLine());
380         assertEquals("<p>", inFile.nextLine());
381         assertEquals("Return to", inFile.nextLine());
382         assertEquals("<a href = index.html>",
    inFile.nextLine());
383         assertEquals("index</a>.", inFile.nextLine());
```

```
384         assertEquals("</p>", inFile.nextLine());
385         assertEquals("</body>", inFile.nextLine());
386         assertEquals("</html>", inFile.nextLine());
387     }
388 
389     @Test
390     public void generateHTMLTest_3() {
391         // multiple links in definition
392         Map<String, String> wordDefinition = new Map1L<>();
393         wordDefinition.add("nick", "the coolest person");
394         wordDefinition.add("coolest", "something awesome");
395         wordDefinition.add("person", "a human");
396         Map<String, String> wordDefinitionExpected = new
    Map1L<>();
397         wordDefinitionExpected.add("nick", "the coolest");
398         wordDefinitionExpected.add("coolest", "something
    awesome");
399         wordDefinitionExpected.add("person", "a human");
400         Queue<String> wordBank = new Queue1L<>();
401         wordBank.enqueue("nick");
402         Queue<String> wordBankExpected = new Queue1L<>();
403         wordBankExpected.enqueue("nick");
404         SimpleWriter out = new
    SimpleWriter1L("generate_HTML_test_2.txt");
405         SimpleWriter outFile = new SimpleWriter1L(
406                 "generate_HTML_subtest_2.txt");
407 
408         SimpleReader in = new
    SimpleReader1L("generate_HTML_test_2.txt");
409 
410         String outputFile = "test";
411         String word = "nick";
412         String wordExpected = "nick";
413         String definition = "the coolest person";
414         String definitionExpected = "the <a href =
    coolest.html>coolest</a> <a href = person.html>person</a>";
415 
416         Glossary.generateHTML(wordDefinitionExpected, word,
    definition, out,
417                 outputFile);
418         assertEquals("<a href = nick.html>", in.nextLine());
419         assertEquals("nick", in.nextLine());
420         assertEquals("</a>", in.nextLine());
421         SimpleReader inFile = new SimpleReader1L(
```

```
422                 outputFile + "/" + word + ".html");
423         assertEquals("<head>", inFile.nextLine());
424         assertEquals("<title>", inFile.nextLine());
425         assertEquals(wordExpected, inFile.nextLine());
426         assertEquals("</title>", inFile.nextLine());
427         assertEquals("</head>", inFile.nextLine());
428         assertEquals("<body>", inFile.nextLine());
429         assertEquals("<h2>", inFile.nextLine());
430         assertEquals("<b>", inFile.nextLine());
431         assertEquals("<i>", inFile.nextLine());
432         assertEquals("<font color = \"red\">" + wordExpected +
    "</font>",
433                 inFile.nextLine());
434         assertEquals("</i>", inFile.nextLine());
435         assertEquals("</b>", inFile.nextLine());
436         assertEquals("</h2>", inFile.nextLine());
437         assertEquals("<blockquote>", inFile.nextLine());
438         assertEquals(definitionExpected, inFile.nextLine());
439         assertEquals("</blockquote>", inFile.nextLine());
440         assertEquals("<hr>", inFile.nextLine());
441         assertEquals("<p>", inFile.nextLine());
442         assertEquals("Return to", inFile.nextLine());
443         assertEquals("<a href = index.html>",
    inFile.nextLine());
444         assertEquals("index</a>.", inFile.nextLine());
445         assertEquals("</p>", inFile.nextLine());
446         assertEquals("</body>", inFile.nextLine());
447         assertEquals("</html>", inFile.nextLine());
448     }
449
450     @Test
451     public void generateWordTest_1() {
452         // test output file - test
453         // normal input - no need to test html outputs because
    will be the same
454         Map<String, String> wordDefinition = new Map1L<>();
455         wordDefinition.add("nick", "the coolest");
456         Map<String, String> wordDefinitionExpected = new
    Map1L<>();
457         wordDefinitionExpected.add("nick", "the coolest");
458         Queue<String> wordBank = new Queue1L<>();
459         wordBank.enqueue("nick");
460         Queue<String> wordBankExpected = new Queue1L<>();
461         wordBankExpected.enqueue("nick");
```

```
462         SimpleWriter out = new
    SimpleWriter1L("generate_word_test_1.txt");
463         SimpleReader in = new
    SimpleReader1L("generate_word_test_1.txt");
464         String outputFile = "test";
465         Glossary.generateWord(wordBank, wordDefinition, out,
    outputFile);
466         assertEquals(wordDefinitionExpected, wordDefinition);
467         assertEquals(wordBankExpected, wordBank);
468         assertEquals("<li>", in.nextLine());
469         assertEquals("<a href = nick.html>", in.nextLine());
470         assertEquals("nick", in.nextLine());
471         assertEquals("</a>", in.nextLine());
472         assertEquals("</li>", in.nextLine());
473
474     }
475
476     @Test
477     public void generateWordTest_2() {
478         // test output file - test2
479         // normal input - no need to test html outputs because
    will be the same
480         Map<String, String> wordDefinition = new Map1L<>();
481         wordDefinition.add("nick", "the coolest");
482         Map<String, String> wordDefinitionExpected = new
    Map1L<>();
483         wordDefinitionExpected.add("nick", "the coolest");
484         Queue<String> wordBank = new Queue1L<>();
485         wordBank.enqueue("nick");
486         Queue<String> wordBankExpected = new Queue1L<>();
487         wordBankExpected.enqueue("nick");
488         SimpleWriter out = new
    SimpleWriter1L("generate_word_test_1.txt");
489         SimpleReader in = new
    SimpleReader1L("generate_word_test_1.txt");
490         String outputFile = "test2";
491         Glossary.generateWord(wordBank, wordDefinition, out,
    outputFile);
492         assertEquals(wordDefinitionExpected, wordDefinition);
493         assertEquals(wordBankExpected, wordBank);
494         assertEquals("<li>", in.nextLine());
495         assertEquals("<a href = nick.html>", in.nextLine());
496         assertEquals("nick", in.nextLine());
497         assertEquals("</a>", in.nextLine());
```

```
498         assertEquals("</li>", in.nextLine());
499 
500     }
501 
502     @Test
503     public void generateWordTest_4() {
504         // testing different word
505         // normal input - no need to test html outputs because
    will be the same
506         Map<String, String> wordDefinition = new Map1L<>();
507         wordDefinition.add("not nick", "the coolest");
508         Map<String, String> wordDefinitionExpected = new
    Map1L<>();
509         wordDefinitionExpected.add("not nick", "the coolest");
510         Queue<String> wordBank = new Queue1L<>();
511         wordBank.enqueue("not nick");
512         Queue<String> wordBankExpected = new Queue1L<>();
513         wordBankExpected.enqueue("not nick");
514         SimpleWriter out = new
    SimpleWriter1L("generate_word_test_1.txt");
515         SimpleReader in = new
    SimpleReader1L("generate_word_test_1.txt");
516         String outputFile = "test";
517         Glossary.generateWord(wordBank, wordDefinition, out,
    outputFile);
518         assertEquals(wordDefinitionExpected, wordDefinition);
519         assertEquals(wordBankExpected, wordBank);
520         assertEquals("<li>", in.nextLine());
521         assertEquals("<a href = not nick.html>",
    in.nextLine());
522         assertEquals("not nick", in.nextLine());
523         assertEquals("</a>", in.nextLine());
524         assertEquals("</li>", in.nextLine());
525 
526     }
527 
528     @Test
529     // match expected - 1 test since method will print out the
    same
530     public void generateIndexTest_1() {
531         SimpleWriter out = new SimpleWriter1L("test_3.txt");
532         Glossary.generateIndex(out);
533         SimpleReader in = new SimpleReader1L("test_3.txt");
534         assertEquals("<html>", in.nextLine());
```

```
535         assertEquals("<head>", in.nextLine());
536         assertEquals("<title>", in.nextLine());
537         assertEquals("Glossary", in.nextLine());
538         assertEquals("</title>", in.nextLine());
539         assertEquals("</head>", in.nextLine());
540         assertEquals("<body>", in.nextLine());
541         assertEquals("<h2>Glossary</h2>", in.nextLine());
542         assertEquals("<hr>", in.nextLine());
543         assertEquals("<h3>Index</h3>", in.nextLine());
544         assertEquals("<ul>", in.nextLine());
545 
546     }
547 
548     @Test
549     // match expected - 1 test since method will print out the
    same
550     public void generateCloserTest_1() {
551         SimpleWriter out = new SimpleWriter1L("test_4.txt");
552         Glossary.generateCloser(out);
553         SimpleReader in = new SimpleReader1L("test_4.txt");
554         assertEquals("</ul>", in.nextLine());
555         assertEquals("</body>", in.nextLine());
556         assertEquals("</html>", in.nextLine());
557     }
558 
559 }
560 
```