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

/**
 * JUnit test fixture for {@code NaturalNumber}'s constructors and kernel
 * methods.
 *
 * @author Charan Nanduri & Evan Frisbie
 *
 */
public abstract class NaturalNumberTest {

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @return the new number
     * @ensures constructorTest = 0
     */
    protected abstract NaturalNumber constructorTest();

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @param i
     *            {@code int} to initialize from
     * @return the new number
     * @requires i >= 0
     * @ensures constructorTest = i
     */

    protected abstract NaturalNumber constructorTest(int i);

    /**
     * Invokes the appropriate {@code NaturalNumber} constructor for the
     * implementation under test and returns the result.
     *
     * @param s
     *            {@code String} to initialize from
     * @return the new number
     * @requires there exists n: NATURAL (s = TO_STRING(n))
```

```
47      * @ensures s = TO_STRING(constructorTest)
48      */
49     protected abstract NaturalNumber constructorTest(String s);
50
51     /**
52      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
53      * implementation under test and returns the result.
54      *
55      * @param n
56      *            {@code NaturalNumber} to initialize from
57      * @return the new number
58      * @ensures constructorTest = n
59      */
60     protected abstract NaturalNumber constructorTest(NaturalNumber
   n);
61
62     /**
63      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
64      * reference implementation and returns the result.
65      *
66      * @return the new number
67      * @ensures constructorRef = 0
68      */
69     protected abstract NaturalNumber constructorRef();
70
71     /**
72      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
73      * reference implementation and returns the result.
74      *
75      * @param i
76      *            {@code int} to initialize from
77      * @return the new number
78      * @requires i >= 0
79      * @ensures constructorRef = i
80      */
81     protected abstract NaturalNumber constructorRef(int i);
82
83     /**
84      * Invokes the appropriate {@code NaturalNumber} constructor
   for the
85      * reference implementation and returns the result.
86      *
```

```
 87      * @param s
 88      *            {@code String} to initialize from
 89      * @return the new number
 90      * @requires there exists n: NATURAL (s = TO_STRING(n))
 91      * @ensures s = TO_STRING(constructorRef)
 92      */
 93     protected abstract NaturalNumber constructorRef(String s);
 94
 95     /**
 96      * Invokes the appropriate {@code NaturalNumber} constructor
    for the
 97      * reference implementation and returns the result.
 98      *
 99      * @param n
100      *            {@code NaturalNumber} to initialize from
101      * @return the new number
102      * @ensures constructorRef = n
103      */
104     protected abstract NaturalNumber constructorRef(NaturalNumber
    n);
105
106     // TODO - add test cases for four constructors, multiplyBy10,
    divideBy10, isZero
107
108     //Constructor Tests
109     @Test
110     public final void testConstructor() {
111         NaturalNumber test = this.constructorTest();
112         NaturalNumber expected = this.constructorRef();
113         assertEquals(expected, test);
114     }
115
116     @Test
117     public final void testIntConstructor() {
118         NaturalNumber test = this.constructorTest(5);
119         NaturalNumber expected = this.constructorRef(5);
120         assertEquals(expected, test);
121     }
122
123     @Test
124     public final void testStringConstructor() {
125         NaturalNumber test = this.constructorTest("9");
126         NaturalNumber expected = this.constructorRef("9");
127         assertEquals(expected, test);
128     }
```

```
129
130     @Test
131     public final void testNaturalNumberConstructor() {
132         NaturalNumber test = this.constructorTest(new
    NaturalNumber1L(5));
133         NaturalNumber expected = this.constructorRef(new
    NaturalNumber1L(5));
134         assertEquals(expected, test);
135     }
136
137     //Kernel tests
138     //Starting with MultiplyBy10 cases
139     @Test
140     public final void testMultiplyBy10Int0() {
141         int original = 0;
142         int addTo = 0;
143         NaturalNumber n = this.constructorTest(original);
144         n.multiplyBy10(addTo);
145         NaturalNumber nnExp = this.constructorTest(original);
146         nnExp.multiplyBy10(addTo);
147         assertEquals(n, nnExp);
148
149     }
150
151     @Test
152     public final void testMultiplyBy10Non0Int() {
153         int original = 2;
154         int addTo = 4;
155         NaturalNumber n = this.constructorTest(original);
156         n.multiplyBy10(addTo);
157         NaturalNumber nnExp = this.constructorRef(original);
158         nnExp.multiplyBy10(addTo);
159         assertEquals(n, nnExp);
160
161     }
162
163     @Test
164     public final void testMultiplyBy10Max() {
165         int maxInt = Integer.MAX_VALUE;
166         NaturalNumber n = this.constructorTest(maxInt);
167         NaturalNumber nnExpected = this.constructorRef(maxInt);
168         nnExpected.multiplyBy10(0);
169         n.multiplyBy10(0);
170         assertEquals(nnExpected, n);
171     }
```

```
172
173     @Test
174     public final void testMultiplyBy10Non0String() {
175         String original = "4";
176         int addTo = 3;
177         NaturalNumber n = this.constructorTest(original);
178         n.multiplyBy10(addTo);
179         NaturalNumber nnExp = this.constructorRef(original);
180         nnExp.multiplyBy10(addTo);
181         assertEquals(n, nnExp);
182     }
183
184     @Test
185     public final void testMultiplyBy10WithMaxIntString() {
186         String max = Integer.toString(Integer.MAX_VALUE);
187         NaturalNumber n = this.constructorTest(max);
188         NaturalNumber nnExp = this.constructorRef(max);
189         nnExp.multiplyBy10(0);
190         n.multiplyBy10(0);
191         assertEquals(nnExp, n);
192     }
193     /*
194      * need to fix
195      *
196      * @Test public final void testMultiplyBy10With0String()
    { String original =
197      * "0"; int addTo = ; // 0 NaturalNumber nn =
198      * this.constructorTest(original); nn.multiplyBy10(); //addTo
    NaturalNumber
199      * nnExp = this.constructorTest(original);
    nnExp.multiplyBy10(addTo);
200      * assertEquals(nn, nnExp); }
201      */
202
203     @Test
204     public final void testMultilyBy10NonZeroNatNum() {
205         NaturalNumber testN = this.constructorTest(3);
206         NaturalNumber refN = this.constructorRef(3);
207         int addTo = 4;
208         NaturalNumber n = this.constructorTest(testN);
209         n.multiplyBy10(addTo);
210         NaturalNumber nnExp = this.constructorRef(refN);
211         nnExp.multiplyBy10(addTo);
212         assertEquals(n, nnExp);
213     }
```

```
214
215     @Test
216     public final void testMultiplyBy10ZeroNatNum() {
217         NaturalNumber testN = this.constructorTest(0);
218         NaturalNumber refN = this.constructorRef(0);
219         int addTo = 0;
220         NaturalNumber n = this.constructorTest(testN);
221         n.multiplyBy10(addTo);
222         NaturalNumber nnExp = this.constructorTest(refN);
223         nnExp.multiplyBy10(addTo);
224         assertEquals(n, nnExp);
225     }
226
227     @Test
228     public final void testMultiplyBy10WithMaxIntsNatNum() {
229         NaturalNumber testN =
    this.constructorTest(Integer.MAX_VALUE);
230         NaturalNumber refN =
    this.constructorRef(Integer.MAX_VALUE);
231         NaturalNumber n = this.constructorTest(testN);
232         NaturalNumber nnExpected = this.constructorRef(refN);
233         nnExpected.multiplyBy10(0);
234         n.multiplyBy10(0);
235         assertEquals(nnExpected, n);
236     }
237
238     //divideBy10 tests
239     @Test
240     public final void testDivideBy10SingleDigWithRInt() {
241         int dividend = 4;
242         NaturalNumber n = this.constructorTest(dividend);
243         int remainder = n.divideBy10();
244         NaturalNumber nnExp = this.constructorRef(dividend);
245         int remainderExp = nnExp.divideBy10();
246         assertEquals(n, nnExp);
247         assertEquals(remainder, remainderExp);
248     }
249
250     @Test
251     public final void testDivideBy10MultipleDigitsWithRInt() {
252         int dividend = 84;
253         NaturalNumber n = this.constructorTest(dividend);
254         int remainder = n.divideBy10();
255         NaturalNumber nnExp = this.constructorRef(dividend);
256         int remainderExp = nnExp.divideBy10();
```

```
257         assertEquals(n, nnExp);
258         assertEquals(remainder, remainderExp);
259     }
260 
261     @Test
262     public final void testDivideBy10SingleDigitNoRInt() {
263         int dividend = 0;
264         NaturalNumber n = this.constructorTest(dividend);
265         int remainder = n.divideBy10();
266         NaturalNumber nnExp = this.constructorRef(dividend);
267         int remainderExp = nnExp.divideBy10();
268         assertEquals(n, nnExp);
269         assertEquals(remainder, remainderExp);
270     }
271 
272     @Test
273     public final void testDivideBy10MultipleDigitsNoRInt() {
274         int dividend = 230;
275         NaturalNumber n = this.constructorTest(dividend);
276         int remainder = n.divideBy10();
277         NaturalNumber nnExp = this.constructorRef(dividend);
278         int remainderExp = nnExp.divideBy10();
279         assertEquals(n, nnExp);
280         assertEquals(remainder, remainderExp);
281     }
282 
283     @Test
284     public final void testDivideBy10SingleDigitWithRStr() {
285         String dividend = "4";
286         NaturalNumber n = this.constructorTest(dividend);
287         int remainder = n.divideBy10();
288         NaturalNumber nnExp = this.constructorRef(dividend);
289         int remainderExp = nnExp.divideBy10();
290         assertEquals(n, nnExp);
291         assertEquals(remainder, remainderExp);
292     }
293 
294     @Test
295     public final void testDivideBy10MultipleDigitsWithRStr() {
296         String dividend = "84";
297         NaturalNumber n = this.constructorTest(dividend);
298         int remainder = n.divideBy10();
299         NaturalNumber nnExp = this.constructorRef(dividend);
300         int remainderExp = nnExp.divideBy10();
301         assertEquals(n, nnExp);
```

```
302         assertEquals(remainder, remainderExp);
303     }
304
305     @Test
306     public final void testDivideBy10SingleDigitWithoutRStr() {
307         String dividend = "0";
308         NaturalNumber n = this.constructorTest(dividend);
309         int remainder = n.divideBy10();
310         NaturalNumber nnExp = this.constructorRef(dividend);
311         int remainderExp = nnExp.divideBy10();
312         assertEquals(n, nnExp);
313         assertEquals(remainder, remainderExp);
314     }
315
316     @Test
317     public final void testDivideBy10MultipleDigitsWithoutRStr() {
318         String dividend = "230";
319         NaturalNumber n = this.constructorTest(dividend);
320         int remainder = n.divideBy10();
321         NaturalNumber nnExp = this.constructorRef(dividend);
322         int remainderExp = nnExp.divideBy10();
323         assertEquals(n, nnExp);
324         assertEquals(remainder, remainderExp);
325     }
326
327     @Test
328     public final void testDivideBy10SingleDigitWithRNatNum() {
329         NaturalNumber testDividend = this.constructorTest(4);
330         NaturalNumber refDividend = this.constructorRef(4);
331         NaturalNumber n = this.constructorTest(testDividend);
332         int remainder = n.divideBy10();
333         NaturalNumber nnExp = this.constructorRef(refDividend);
334         int remainderExp = nnExp.divideBy10();
335         assertEquals(n, nnExp);
336         assertEquals(remainder, remainderExp);
337     }
338
339     @Test
340     public final void testDivideBy10MultipleDigitsWithRNatNum() {
341         NaturalNumber testDividend = this.constructorTest(48);
342         NaturalNumber refDividend = this.constructorRef(48);
343         NaturalNumber n = this.constructorTest(testDividend);
344         int remainder = n.divideBy10();
345         NaturalNumber nnExp = this.constructorRef(refDividend);
346         int remainderExp = nnExp.divideBy10();
```

```
347         assertEquals(n, nnExp);
348         assertEquals(remainder, remainderExp);
349     }
350 
351     @Test
352     public final void testDivideBy10SingleDigitNoRemainderNatNum()
    {
353         NaturalNumber testDividend = this.constructorTest(0);
354         NaturalNumber refDividend = this.constructorRef(0);
355         NaturalNumber n = this.constructorTest(testDividend);
356         int remainder = n.divideBy10();
357         NaturalNumber nnExp = this.constructorRef(refDividend);
358         int remainderExp = nnExp.divideBy10();
359         assertEquals(n, nnExp);
360         assertEquals(remainder, remainderExp);
361     }
362 
363     @Test
364     public final void testDivideBy10MultipleDigitsNoRNatNum() {
365         NaturalNumber testDividend = this.constructorTest(230);
366         NaturalNumber refDividend = this.constructorRef(230);
367         NaturalNumber n = this.constructorTest(testDividend);
368         int remainder = n.divideBy10();
369         NaturalNumber nnExp = this.constructorRef(refDividend);
370         int remainderExp = nnExp.divideBy10();
371         assertEquals(n, nnExp);
372         assertEquals(remainder, remainderExp);
373     }
374     //isZero tests
375 
376     @Test
377     public final void testIsZeroFalse() {
378         int num = 43;
379         NaturalNumber n = this.constructorTest(num);
380         NaturalNumber nnExp = this.constructorRef(num);
381         assertEquals(n.isZero(), nnExp.isZero());
382     }
383 
384     @Test
385     public final void testIsZeroTrue() {
386         NaturalNumber n = this.constructorTest();
387         NaturalNumber nnExp = this.constructorRef();
388         assertEquals(n.isZero(), nnExp.isZero());
389     }
390 
```

```
391 }
392
```