```
See the Assessment Guide for information on how to interpret
this report.

ASSESSMENT SUMMARY

Compilation:  FAILED (0 errors, 5 warnings)
API:          PASSED

Findbugs:     PASSED
Checkstyle:   FAILED (44 warnings)

Correctness:  29/43 tests passed
Memory:       50/53 tests passed
Timing:       57/110 tests passed

Aggregate score: 65.26%
[Compilation: 5%, API: 5%, Findbugs: 0%, Checkstyle: 0%,
Correctness: 60%, Memory: 10%, Timing: 20%]

ASSESSMENT DETAILS

The following files were submitted:
----------------------------------
3.4K Jun 11 11:29 Deque\ copy.java
3.4K Jun 11 11:29 Deque.java
 487 Jun 11 11:29 Permutation\ copy.java
 487 Jun 11 11:29 Permutation.java
3.6K Jun 11 11:29 RandomizedQueue\ copy.java
3.6K Jun 11 11:29 RandomizedQueue.java


*************************************************************
*****************
*  COMPILING
*************************************************************
*****************


% javac Deque.java
*-----------------------------------------------------------

% javac RandomizedQueue.java
*-----------------------------------------------------------
RandomizedQueue.java:17: warning: [unchecked] unchecked cast
        list = (Item[]) new Object[2];
                        ^
  required: Item[]
  found:    Object[]
  where Item is a type-variable:
    Item extends Object declared in class RandomizedQueue
```

```
RandomizedQueue.java:28: warning: [unchecked] unchecked cast
        newlist = (Item[]) new Object[newsize];
                           ^
  required: Item[]
  found:    Object[]
  where Item is a type-variable:
    Item extends Object declared in class RandomizedQueue
2 warnings

% javac Permutation.java
*---------------------------------------------------------
Permutation.java:7: warning: [rawtypes] found raw type:
RandomizedQueue
        RandomizedQueue randque = new RandomizedQueue();
        ^
  missing type arguments for generic class
RandomizedQueue<Item>
  where Item is a type-variable:
    Item extends Object declared in class RandomizedQueue
Permutation.java:7: warning: [rawtypes] found raw type:
RandomizedQueue
        RandomizedQueue randque = new RandomizedQueue();
                                      ^
  missing type arguments for generic class
RandomizedQueue<Item>
  where Item is a type-variable:
    Item extends Object declared in class RandomizedQueue
Permutation.java:10: warning: [unchecked] unchecked call to
enqueue(Item) as a member of the raw type RandomizedQueue
            randque.enqueue(item);
                            ^
  where Item is a type-variable:
    Item extends Object declared in class RandomizedQueue
3 warnings


================================================================
==


Checking the APIs of your programs.
*-----------------------------------------------------------
Deque:

RandomizedQueue:

Permutation:

================================================================
==
```

```
*****************************************************************
******************
*  CHECKING STYLE AND COMMON BUG PATTERNS
*****************************************************************
******************


% findbugs *.class
*---------------------------------------------------------------


================================================================
==


% checkstyle *.java
*---------------------------------------------------------------
Deque.java:17:18: '{' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:22:28: '{' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:25:21: '{' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:33:10: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:34:17: '==' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:34:19: '==' is not followed by whitespace.
[WhitespaceAround]
Deque.java:42:10: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:42:18: '!=' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:43:10: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:46:29: '{' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:47:10: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:62:36: '{' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:74:3: The comment is empty. [IllegalTokenText]
Deque.java:99:10: 'for' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:99:33: '{' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:100:13: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:100:33: ')' is preceded with whitespace. [ParenPad]
```

```
Deque.java:101:13: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:101:33: ')' is preceded with whitespace. [ParenPad]
Deque.java:102:13: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:102:59: '{' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:103:17: 'if' is not followed by whitespace.
[WhitespaceAfter]
Deque.java:103:19: '%' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:103:20: '%' is not followed by whitespace.
[WhitespaceAround]
Deque.java:103:21: '==' is not preceded with whitespace.
[WhitespaceAround]
Deque.java:103:23: '==' is not followed by whitespace.
[WhitespaceAround]
RandomizedQueue.java:4:8: Unused import statement for
'edu.princeton.cs.algs4.StdIn'. [UnusedImports]
RandomizedQueue.java:7:8: Unused import statement for
'edu.princeton.cs.algs4.StdOut'. [UnusedImports]
RandomizedQueue.java:27:37: '{' is not preceded with
whitespace. [WhitespaceAround]
RandomizedQueue.java:29:12: 'for' is not followed by
whitespace. [WhitespaceAfter]
RandomizedQueue.java:29:39: '{' is not preceded with
whitespace. [WhitespaceAround]
RandomizedQueue.java:35:35: '{' is not preceded with
whitespace. [WhitespaceAround]
RandomizedQueue.java:42:11: 'if' is not followed by
whitespace. [WhitespaceAfter]
RandomizedQueue.java:56:37: '{' is not preceded with
whitespace. [WhitespaceAround]
RandomizedQueue.java:63:36: '{' is not preceded with
whitespace. [WhitespaceAround]
RandomizedQueue.java:67:16: 'for' is not followed by
whitespace. [WhitespaceAfter]
RandomizedQueue.java:67:39: '{' is not preceded with
whitespace. [WhitespaceAround]
RandomizedQueue.java:68:27: '=' is not followed by whitespace.
[WhitespaceAround]
RandomizedQueue.java:73:13: Conditional logic can be removed.
[SimplifyBooleanReturn]
...
 Checkstyle ends with 44 errors.

================================================================
==
```

```
************************************************************
*****************
*  TESTING CORRECTNESS
************************************************************
*****************

Testing correctness of Deque
*-----------------------------------------------------------
Running 16 total tests.

Tests 1-6 make random calls to addFirst(), addLast(),
removeFirst(),
removeLast(), isEmpty(), and size(). The probabilities of each
operation are (p1, p2, p3, p4, p5, p6), respectively.

Test 1: Calls to addFirst(), addLast(), and size()
  *    5 random calls (0.4, 0.4, 0.0, 0.0, 0.0, 0.2)
  *   50 random calls (0.4, 0.4, 0.0, 0.0, 0.0, 0.2)
  *  500 random calls (0.4, 0.4, 0.0, 0.0, 0.0, 0.2)
  * 1000 random calls (0.4, 0.4, 0.0, 0.0, 0.0, 0.2)
==> passed

Test 2: Calls to addFirst(), removeFirst(), and isEmpty()
  *    5 random calls (0.8, 0.0, 0.1, 0.0, 0.1, 0.0)
  *   50 random calls (0.8, 0.0, 0.1, 0.0, 0.1, 0.0)
  *  500 random calls (0.8, 0.0, 0.1, 0.0, 0.1, 0.0)
  * 1000 random calls (0.8, 0.0, 0.1, 0.0, 0.1, 0.0)
  *    5 random calls (0.1, 0.0, 0.8, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test2(TestDeque.java:168)
    TestDeque.main(TestDeque.java:740)

    - sequence of dequeue operations was:
        deque.isEmpty()
        deque.isEmpty()
        deque.addFirst(2)
        deque.removeFirst()
  *   50 random calls (0.1, 0.0, 0.8, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test2(TestDeque.java:169)
    TestDeque.main(TestDeque.java:740)
```

```
    - sequence of dequeue operations was:
          deque.addFirst(0)
          deque.removeFirst()
  *  500 random calls (0.1, 0.0, 0.8, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test2(TestDeque.java:170)
    TestDeque.main(TestDeque.java:740)

    - sequence of dequeue operations was:
          deque.addFirst(0)
          deque.removeFirst()
  * 1000 random calls (0.1, 0.0, 0.8, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test2(TestDeque.java:171)
    TestDeque.main(TestDeque.java:740)

    - sequence of dequeue operations was:
          deque.addFirst(0)
          deque.removeFirst()
==> FAILED

Test 3: Calls to addFirst(), removeLast(), and isEmpty()
  *    5 random calls (0.8, 0.0, 0.0, 0.1, 0.1, 0.0)
  *   50 random calls (0.8, 0.0, 0.0, 0.1, 0.1, 0.0)
  *  500 random calls (0.8, 0.0, 0.0, 0.1, 0.1, 0.0)
  * 1000 random calls (0.8, 0.0, 0.0, 0.1, 0.1, 0.0)
  *    5 random calls (0.1, 0.0, 0.0, 0.8, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$202(Deque.java:11)
    Deque.removeLast(Deque.java:58)
    TestDeque.random(TestDeque.java:87)
    TestDeque.test3(TestDeque.java:184)
    TestDeque.main(TestDeque.java:741)

    - sequence of dequeue operations was:
          deque.isEmpty()
          deque.isEmpty()
          deque.isEmpty()
          deque.addFirst(3)
          deque.removeLast()
  *   50 random calls (0.1, 0.0, 0.0, 0.8, 0.1, 0.0)
```

```
    java.lang.NullPointerException

    Deque$DequeNode.access$202(Deque.java:11)
    Deque.removeLast(Deque.java:58)
    TestDeque.random(TestDeque.java:87)
    TestDeque.test3(TestDeque.java:185)
    TestDeque.main(TestDeque.java:741)

    - sequence of dequeue operations was:
         deque.isEmpty()
         deque.isEmpty()
         deque.addFirst(2)
         deque.isEmpty()
         deque.isEmpty()
         deque.addFirst(5)
         deque.removeLast()      ==> 2
         deque.removeLast()
  *  500 random calls (0.1, 0.0, 0.0, 0.8, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$202(Deque.java:11)
    Deque.removeLast(Deque.java:58)
    TestDeque.random(TestDeque.java:87)
    TestDeque.test3(TestDeque.java:186)
    TestDeque.main(TestDeque.java:741)

    - sequence of dequeue operations was:
         deque.addFirst(0)
         deque.addFirst(1)
         deque.addFirst(2)
         deque.removeLast()      ==> 0
         deque.removeLast()      ==> 1
         deque.removeLast()
  * 1000 random calls (0.1, 0.0, 0.0, 0.8, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$202(Deque.java:11)
    Deque.removeLast(Deque.java:58)
    TestDeque.random(TestDeque.java:87)
    TestDeque.test3(TestDeque.java:187)
    TestDeque.main(TestDeque.java:741)

    - sequence of dequeue operations was:
         deque.addFirst(0)
         deque.removeLast()
==> FAILED

Test 4: Calls to addLast(), removeLast(), and isEmpty()
  *    5 random calls (0.0, 0.8, 0.0, 0.1, 0.1, 0.0)
  *   50 random calls (0.0, 0.8, 0.0, 0.1, 0.1, 0.0)
```

```
*  500 random calls (0.0, 0.8, 0.0, 0.1, 0.1, 0.0)
  java.lang.NullPointerException

  Deque$DequeNode.access$202(Deque.java:11)
  Deque.removeLast(Deque.java:58)
  TestDeque.random(TestDeque.java:87)
  TestDeque.test4(TestDeque.java:198)
  TestDeque.main(TestDeque.java:742)

  - sequence of dequeue operations was:
       deque.isEmpty()
       deque.addLast(1)
       deque.removeLast()
* 1000 random calls (0.0, 0.8, 0.0, 0.1, 0.1, 0.0)
*    5 random calls (0.0, 0.1, 0.0, 0.8, 0.1, 0.0)
  java.lang.NullPointerException

  Deque$DequeNode.access$202(Deque.java:11)
  Deque.removeLast(Deque.java:58)
  TestDeque.random(TestDeque.java:87)
  TestDeque.test4(TestDeque.java:200)
  TestDeque.main(TestDeque.java:742)

  - sequence of dequeue operations was:
       deque.addLast(0)
       deque.removeLast()
*   50 random calls (0.0, 0.1, 0.0, 0.8, 0.1, 0.0)
  java.lang.NullPointerException

  Deque$DequeNode.access$202(Deque.java:11)
  Deque.removeLast(Deque.java:58)
  TestDeque.random(TestDeque.java:87)
  TestDeque.test4(TestDeque.java:201)
  TestDeque.main(TestDeque.java:742)

  - sequence of dequeue operations was:
       deque.isEmpty()
       deque.isEmpty()
       deque.isEmpty()
       deque.addLast(3)
       deque.removeLast()
*  500 random calls (0.0, 0.1, 0.0, 0.8, 0.1, 0.0)
  java.lang.NullPointerException

  Deque$DequeNode.access$202(Deque.java:11)
  Deque.removeLast(Deque.java:58)
  TestDeque.random(TestDeque.java:87)
  TestDeque.test4(TestDeque.java:202)
  TestDeque.main(TestDeque.java:742)
```

```
    - sequence of dequeue operations was:
          deque.isEmpty()
          deque.addLast(1)
          deque.removeLast()
  * 1000 random calls (0.0, 0.1, 0.0, 0.8, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$202(Deque.java:11)
    Deque.removeLast(Deque.java:58)
    TestDeque.random(TestDeque.java:87)
    TestDeque.test4(TestDeque.java:203)
    TestDeque.main(TestDeque.java:742)

    - sequence of dequeue operations was:
          deque.addLast(0)
          deque.isEmpty()
          deque.removeLast()
==> FAILED

Test 5: Calls to addLast(), removeFirst(), and isEmpty()
  *    5 random calls (0.0, 0.8, 0.1, 0.0, 0.1, 0.0)
  *   50 random calls (0.0, 0.8, 0.1, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test5(TestDeque.java:213)
    TestDeque.main(TestDeque.java:743)

    - sequence of dequeue operations was:
          deque.addLast(0)
          deque.removeFirst()
  *  500 random calls (0.0, 0.8, 0.1, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test5(TestDeque.java:214)
    TestDeque.main(TestDeque.java:743)

    - sequence of dequeue operations was:
          deque.addLast(0)
          deque.isEmpty()
          deque.removeFirst()
  * 1000 random calls (0.0, 0.8, 0.1, 0.0, 0.1, 0.0)
  *    5 random calls (0.0, 0.1, 0.8, 0.0, 0.1, 0.0)
  *   50 random calls (0.0, 0.1, 0.8, 0.0, 0.1, 0.0)
    java.lang.NullPointerException
```

```
    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test5(TestDeque.java:217)
    TestDeque.main(TestDeque.java:743)

    - sequence of dequeue operations was:
         deque.isEmpty()
         deque.isEmpty()
         deque.isEmpty()
         deque.isEmpty()
         deque.addLast(4)
         deque.removeFirst()
  *  500 random calls (0.0, 0.1, 0.8, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test5(TestDeque.java:218)
    TestDeque.main(TestDeque.java:743)

    - sequence of dequeue operations was:
         deque.isEmpty()
         deque.isEmpty()
         deque.isEmpty()
         deque.addLast(3)
         deque.removeFirst()
  * 1000 random calls (0.0, 0.1, 0.8, 0.0, 0.1, 0.0)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test5(TestDeque.java:219)
    TestDeque.main(TestDeque.java:743)

    - sequence of dequeue operations was:
         deque.isEmpty()
         deque.addLast(1)
         deque.removeFirst()
==> FAILED

Test 6: Calls to addFirst(), addLast(), removeFirst(),
        removeLast(), isEmpty(), and size().
  *    5 random calls (0.3, 0.3, 0.1, 0.1, 0.1, 0.1)
  *   50 random calls (0.3, 0.3, 0.1, 0.1, 0.1, 0.1)
    java.lang.NullPointerException
```

```
  Deque$DequeNode.access$202(Deque.java:11)
  Deque.removeLast(Deque.java:58)
  TestDeque.random(TestDeque.java:87)
  TestDeque.test6(TestDeque.java:231)
  TestDeque.main(TestDeque.java:744)

  - sequence of dequeue operations was:
       deque.addLast(0)
       deque.removeLast()
*  500 random calls (0.3, 0.3, 0.1, 0.1, 0.1, 0.1)
  java.lang.NullPointerException

  Deque$DequeNode.access$202(Deque.java:11)
  Deque.removeLast(Deque.java:58)
  TestDeque.random(TestDeque.java:87)
  TestDeque.test6(TestDeque.java:232)
  TestDeque.main(TestDeque.java:744)

  - sequence of dequeue operations was:
       deque.isEmpty()
       deque.addFirst(1)
       deque.removeLast()
* 1000 random calls (0.3, 0.3, 0.1, 0.1, 0.1, 0.1)
*    5 random calls (0.1, 0.1, 0.3, 0.3, 0.1, 0.1)
  java.lang.NullPointerException

  Deque$DequeNode.access$302(Deque.java:11)
  Deque.removeFirst(Deque.java:50)
  TestDeque.random(TestDeque.java:69)
  TestDeque.test6(TestDeque.java:234)
  TestDeque.main(TestDeque.java:744)

  - sequence of dequeue operations was:
       deque.isEmpty()
       deque.size()
       deque.isEmpty()
       deque.addLast(3)
       deque.removeFirst()
*   50 random calls (0.1, 0.1, 0.3, 0.3, 0.1, 0.1)
  java.lang.NullPointerException

  Deque$DequeNode.access$202(Deque.java:11)
  Deque.removeLast(Deque.java:58)
  TestDeque.random(TestDeque.java:87)
  TestDeque.test6(TestDeque.java:235)
  TestDeque.main(TestDeque.java:744)

  - sequence of dequeue operations was:
       deque.addFirst(0)
       deque.isEmpty()
```

```
         deque.isEmpty()
         deque.isEmpty()
         deque.addFirst(4)
         deque.removeFirst()     ==> 4
         deque.removeLast()
  *  500 random calls (0.1, 0.1, 0.3, 0.3, 0.1, 0.1)
    java.lang.NullPointerException

    Deque$DequeNode.access$202(Deque.java:11)
    Deque.removeLast(Deque.java:58)
    TestDeque.random(TestDeque.java:87)
    TestDeque.test6(TestDeque.java:236)
    TestDeque.main(TestDeque.java:744)

    - sequence of dequeue operations was:
         deque.addFirst(0)
         deque.isEmpty()
         deque.removeLast()
  * 1000 random calls (0.1, 0.1, 0.3, 0.3, 0.1, 0.1)
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.random(TestDeque.java:69)
    TestDeque.test6(TestDeque.java:237)
    TestDeque.main(TestDeque.java:744)

    - sequence of dequeue operations was:
         deque.isEmpty()
         deque.addLast(1)
         deque.size()
         deque.removeFirst()
==> FAILED

Test 7: Removing from an empty deque
  * removeFirst()
  * removeLast()
     java.util.NoSuchElementException not thrown
==> FAILED

Test 8: Create multiple deque objects at the same time
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.twoDeques(TestDeque.java:303)
    TestDeque.test8(TestDeque.java:337)
    TestDeque.main(TestDeque.java:746)

    java.lang.NullPointerException
```

```
    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.twoDeques(TestDeque.java:303)
    TestDeque.test8(TestDeque.java:338)
    TestDeque.main(TestDeque.java:746)

==> FAILED

Test 9: Check iterator() after calls only to addFirst()
==> passed

Test 10: Check iterator() after intermixed calls to
addFirst(), addLast(),
         removeFirst(), and removeLast()
    java.lang.NullPointerException

    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.test10(TestDeque.java:391)
    TestDeque.main(TestDeque.java:748)

    - sequence of dequeue operations was:
          deque.addFirst(1)
          deque.addLast(2)
          deque.removeLast()    ==> 2
          deque.addLast(4)
          deque.removeLast()    ==> 4
          deque.addFirst(6)
          deque.removeFirst()   ==> 6
          deque.addLast(8)
          deque.addLast(9)
          deque.removeFirst()   ==> 1
          deque.removeLast()    ==> 9
          deque.removeFirst()
==> FAILED

Test 11: Create two nested iterators to same deque
  * n = 10
  * n = 1000
==> passed

Test 12: Create two parallel iterators to same deque
  * n = 10
  * n = 1000
==> passed

Test 13: Create Deque objects of different parameterized types
    java.lang.NullPointerException
```

```
    Deque$DequeNode.access$302(Deque.java:11)
    Deque.removeFirst(Deque.java:50)
    TestDeque.test13(TestDeque.java:613)
    TestDeque.main(TestDeque.java:751)

==> FAILED

Test 14: Check that addFirst() and addLast() each throw a
NullPointerException
         when inserting null items
    - java.lang.NullPointerException not thrown for
addFirst(null)
    - java.lang.NullPointerException not thrown for
addLast(null)
==> FAILED

Test 15: Check that remove() and next() throw the specified
exceptions in iterator()
==> passed

Test 16: Check iterator() when Deque is empty
==> passed


Total: 6/16 tests passed!


================================================================
==
Testing correctness of RandomizedQueue
*----------------------------------------------------------
Running 18 total tests.

Tests 1-4 make random calls to enqueue(), dequeue(), sample(),
isEmpty(), and size(). The probabilities of each operation are
(p1, p2, p3, p4, p5), respectively.

Test 1: check random calls to enqueue() and size()
  *    5 random calls (0.8, 0.0, 0.0, 0.0, 0.2)
  *   50 random calls (0.8, 0.0, 0.0, 0.0, 0.2)
  *  500 random calls (0.8, 0.0, 0.0, 0.0, 0.2)
  * 1000 random calls (0.8, 0.0, 0.0, 0.0, 0.2)
==> passed

Test 2: check random calls to enqueue() and dequeue()
  *    5 random calls (0.7, 0.1, 0.0, 0.1, 0.1)
  *   50 random calls (0.7, 0.1, 0.0, 0.1, 0.1)
  *  500 random calls (0.7, 0.1, 0.0, 0.1, 0.1)
  * 1000 random calls (0.7, 0.1, 0.0, 0.1, 0.1)
  *    5 random calls (0.1, 0.7, 0.0, 0.1, 0.1)
```

```
  *   50 random calls (0.1, 0.7, 0.0, 0.1, 0.1)
    java.lang.ArrayIndexOutOfBoundsException: 0

    RandomizedQueue.enqueue(RandomizedQueue.java:38)
    TestRandomizedQueue.random(TestRandomizedQueue.java:82)
    TestRandomizedQueue.test2(TestRandomizedQueue.java:214)
    TestRandomizedQueue.main(TestRandomizedQueue.java:1015)

    - sequence of dequeue operations was:
        rq.enqueue(15)
        rq.size()        ==> 1
        rq.dequeue()     ==> 15
        rq.enqueue(42)
        rq.dequeue()     ==> 42
        rq.isEmpty()     ==> true
        rq.size()        ==> 0
        rq.isEmpty()     ==> true
        rq.enqueue(24)

  *  500 random calls (0.1, 0.7, 0.0, 0.1, 0.1)
    java.lang.ArrayIndexOutOfBoundsException: 0

    RandomizedQueue.enqueue(RandomizedQueue.java:38)
    TestRandomizedQueue.random(TestRandomizedQueue.java:82)
    TestRandomizedQueue.test2(TestRandomizedQueue.java:215)
    TestRandomizedQueue.main(TestRandomizedQueue.java:1015)

  * 1000 random calls (0.1, 0.7, 0.0, 0.1, 0.1)
    java.lang.ArrayIndexOutOfBoundsException: 0

    RandomizedQueue.enqueue(RandomizedQueue.java:38)
    TestRandomizedQueue.random(TestRandomizedQueue.java:82)
    TestRandomizedQueue.test2(TestRandomizedQueue.java:216)
    TestRandomizedQueue.main(TestRandomizedQueue.java:1015)

==> FAILED

Test 3: check random calls to enqueue(), sample(), and size()
  *    5 random calls (0.8, 0.0, 0.1, 0.0, 0.1)
  *   50 random calls (0.8, 0.0, 0.1, 0.0, 0.1)
  *  500 random calls (0.8, 0.0, 0.1, 0.0, 0.1)
  * 1000 random calls (0.8, 0.0, 0.1, 0.0, 0.1)
  *    5 random calls (0.1, 0.0, 0.8, 0.0, 0.1)
  *   50 random calls (0.1, 0.0, 0.8, 0.0, 0.1)
  *  500 random calls (0.1, 0.0, 0.8, 0.0, 0.1)
  * 1000 random calls (0.1, 0.0, 0.8, 0.0, 0.1)
==> passed

Test 4: check random calls to enqueue(), dequeue(), sample(),
isEmpty(), and size()
```

```
  *    5 random calls (0.6, 0.1, 0.1, 0.1, 0.1)
  *   50 random calls (0.6, 0.1, 0.1, 0.1, 0.1)
  *  500 random calls (0.6, 0.1, 0.1, 0.1, 0.1)
  * 1000 random calls (0.6, 0.1, 0.1, 0.1, 0.1)
  *    5 random calls (0.1, 0.1, 0.6, 0.1, 0.1)
  *   50 random calls (0.1, 0.1, 0.6, 0.1, 0.1)
  *  500 random calls (0.1, 0.1, 0.6, 0.1, 0.1)
    java.lang.ArrayIndexOutOfBoundsException: 0

    RandomizedQueue.enqueue(RandomizedQueue.java:38)
    TestRandomizedQueue.random(TestRandomizedQueue.java:82)
    TestRandomizedQueue.test4(TestRandomizedQueue.java:243)
    TestRandomizedQueue.main(TestRandomizedQueue.java:1017)

    - sequence of dequeue operations was:
        rq.enqueue(450)
        rq.isEmpty()     ==> false
        rq.dequeue()     ==> 450
        rq.isEmpty()     ==> true
        rq.size()        ==> 0
        rq.isEmpty()     ==> true
        rq.enqueue(201)
        rq.sample()      ==> 201
        rq.sample()      ==> 201
        rq.dequeue()     ==> 201
        rq.enqueue(86)

  * 1000 random calls (0.1, 0.1, 0.6, 0.1, 0.1)
    java.lang.ArrayIndexOutOfBoundsException: 0

    RandomizedQueue.enqueue(RandomizedQueue.java:38)
    TestRandomizedQueue.random(TestRandomizedQueue.java:82)
    TestRandomizedQueue.test4(TestRandomizedQueue.java:244)
    TestRandomizedQueue.main(TestRandomizedQueue.java:1017)

==> FAILED

Test 5: call dequeue() and sample() from an empty randomized
queue
  * dequeue()
  * sample()
     java.util.NoSuchElementException not thrown
==> FAILED

Test 6: create multiple randomized queue objects at the same
time
==> passed

Test 7: check that iterator() returns correct items after a
sequence of
```

```
        enqueue() operations
==> passed

Test 8: check that iterator() returns correct items after
sequence of enqueue()
        and dequeue() operations
    java.lang.ArrayIndexOutOfBoundsException: 0

    RandomizedQueue.enqueue(RandomizedQueue.java:38)
    TestRandomizedQueue.test8(TestRandomizedQueue.java:391)
    TestRandomizedQueue.main(TestRandomizedQueue.java:1021)

==> FAILED

Test 9: create two nested iterators over the same randomized
queue
  * n = 10
  * n = 1000
==> passed

Test 10: create two parallel iterators over the same
randomized queue
  * n = 10
  * n = 1000
==> passed

Test 11: create two iterators over different randomized queues
==> passed

Test 12: create RandomizedQueue objects of different
parameterized types
==> passed

Test 13: check randomness of sample() by enqueueing n items,
repeatedly calling
         sample(), and counting the frequency of each item
  * n = 3, trials = 12000
  * n = 5, trials = 12000
  * n = 8, trials = 12000
  * n = 10, trials = 12000
==> passed

Test 14: check randomness of dequeue() by enqueueing n items,
dequeueing n items,
         and seeing whether each of the n! permutations is
equally likely
  * n = 2, trials = 12000
  * n = 3, trials = 12000
  * n = 4, trials = 12000
  * n = 5, trials = 12000
```

```
==> passed

Test 15: check randomness of iterator() by enqueueing n items,
iterating over those
         n items, and seeing whether each of the n!
permutations is equally likely
  * n = 2, trials = 12000
  * n = 3, trials = 12000
  * n = 4, trials = 12000
  * n = 5, trials = 12000
==> passed

Test 16: check that NullPointerException is thrown when
inserting null items
==> passed

Test 17: check that remove() and next() throw the specified
exceptions in iterator()
==> passed

Test 18: check iterator() when RandomizedQueue is empty
==> passed


Total: 14/18 tests passed!


================================================================
==
*****************************************************************
*****************
*  TESTING CORRECTNESS (substituting reference RandomizedQueue
and Deque)
*****************************************************************
*****************

Testing correctness of Permutation
*-----------------------------------------------------------
Tests 1-5 call the main() function directly, resetting
standard input
before each call.

Running 9 total tests.

Test 1a: check formatting for sample inputs from assignment
specification
  % java Permutation 3 < distinct.txt
  B
  E
  D
```

```
  % java Permutation 3 < distinct.txt
  D
  A
  H

  % java Permutation 8 < duplicates.txt
  CC
  BB
  AA
  BB
  BB
  BB
  BB
  CC

==> passed

Test 1b: check formatting for other inputs
  % java Permutation 8 < mediumTale.txt
  was
  foolishness
  it
  was
  age
  of
  it
  it

  % java Permutation 0 < distinct.txt
  [no output]

==> passed

Test 2: check that main() reads all data from standard input
  * filename = distinct.txt, k = 3
  * filename = distinct.txt, k = 3
  * filename = duplicates.txt, k = 8
  * filename = mediumTale.txt, k = 8
==> passed

Test 3a: check that main() prints each item from the sequence
at most once
         (for inputs with no duplicate strings)
  * filename = distinct.txt, k = 3
  * filename = distinct.txt, k = 1
  * filename = distinct.txt, k = 9
  * filename = permutation6.txt, k = 6
  * filename = permutation10.txt, k = 10
==> passed
```

```
Test 3b: check that main() prints each item from the sequence
at most once
         (for inputs with duplicate strings)
  * filename = duplicates.txt, k = 8
  * filename = duplicates.txt, k = 3
  * filename = permutation8.txt, k = 6
  * filename = permutation8.txt, k = 2
  * filename = tinyTale.txt, k = 10
==> passed

Test 3c: check that main() prints each item from the sequence
at most once
         (for inputs with newlines)
  * filename = mediumTale.txt, k = 10
  * filename = mediumTale.txt, k = 20
  * filename = tale.txt, k = 10
  * filename = tale.txt, k = 50
==> passed

Test 4: check main() when k = 0
  * filename = distinct.txt, k = 0
  * filename = distinct.txt, k = 0
==> passed

Test 5a: check that permutations are uniformly random
         (for inputs with no duplicate strings)
  * filename = per

...

WARNING: the grading output was truncated due to excessive
length.
Typically, this is because you have a method that has an
unanticipated side effect
(such as printing to standard output or throwing an
exception). A large amount of output
can also arise from failing many tests.
```