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

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     PASSED
PMD:          PASSED
Checkstyle:   PASSED

Correctness:  73/73 tests passed
Memory:       10/10 tests passed
Timing:       163/163 tests passed

Aggregate score: 100.00%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 60%, Timing: 10%, Memory: 20% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
2.4K Dec 16 11:41 BurrowsWheeler.java
1.7K Dec 16 11:41 CircularSuffixArray.java
3.5K Dec 16 11:41 MoveToFront.java


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


% javac CircularSuffixArray.java
*-----------------------------------------------------------

% javac BurrowsWheeler.java
*-----------------------------------------------------------

% javac MoveToFront.java
*-----------------------------------------------------------


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


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

BurrowsWheeler:

MoveToFront:

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


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


% spotbugs *.class
*-----------------------------------------------------------
```

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


% pmd .
*-----------------------------------------------------------



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


% checkstyle *.java
*-----------------------------------------------------------

% custom checkstyle checks for CircularSuffixArray.java
*-----------------------------------------------------------

% custom checkstyle checks for BurrowsWheeler.java
*-----------------------------------------------------------

% custom checkstyle checks for MoveToFront.java
*-----------------------------------------------------------


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


********************************************************************************
*  TESTING CORRECTNESS
********************************************************************************

Testing correctness of CircularSuffixArray
*-----------------------------------------------------------
Running 20 total tests.

Test 1: check index() and length() with strings from text files
  * abra.txt
  * weekend.txt
  * banana.txt
==> passed

Test 2: check index() and length() with random binary strings
  * length = 3
  * length = 4
  * length = 5
  * length = 6
  * length = 7
  * length = 8
  * length = 9
  * length = 10
==> passed

Test 3: check index() and length() with random binary strings
  * length = 50
  * length = 100
  * length = 1000
==> passed

Test 4: check index() and length() with random DNA strings
  * length = 3
  * length = 4
  * length = 5
  * length = 6
  * length = 7
  * length = 8
  * length = 9
  * length = 10
==> passed

Test 5: check index() and length() with random uppercase strings
  * length = 3
  * length = 6
  * length = 10
  * length = 100
  * length = 1000
==> passed

Test 6: check index() and length() with random ASCII strings (excluding 0x00)
  * length = 4
  * length = 7
  * length = 10
  * length = 100
  * length = 1000
==> passed

Test 7: check index() and length() with random ASCII strings
  * length = 5
  * length = 8
  * length = 10
  * length = 100
  * length = 1000
==> passed

Test 8: check index() and length() with random extended ASCII strings
        (excluding 0xFF)
  * length = 10
  * length = 100
  * length = 1000
==> passed

Test 9: check index() and length() with random extended ASCII strings
  * length = 10
  * length = 100
  * length = 1000
==> passed

Test 10: check index() and length() with strings from text files
  * cadabra.txt
  * amendments.txt
  * moby1.txt
  * dickens1000.txt
==> passed

Test 11: check index() and length() with strings from binary files
  * us.gif
  * CS_bricks.jpg
  * rand1K.bin
==> passed

Test 12: check index() and length() with random strings of length 0, 1, and 2
  * length = 0
  * length = 1
  * length = 2
==> passed

Test 13: check that index() throws an exception when argument is out of bounds
  * string of length 10
  * string of length 100
  * string of length 2
  * string of length 1
  * string of length 0
==> passed

Test 14: check that constructor throws an exception when argument is null
==> passed

Test 15: check that two CircularSuffixArray objects can be created at the same time
  * cadabra.txt and amendments.txt
  * amendments.txt and cadabra.txt
  * dickens1000.txt and cadabra.txt
==> passed

Test 16: check that CircularSuffixArray is immutable
  * string = "MZVYLTDKJSLJQICVVOOQXNLQZLJIVD"
  * string = "ABABBBAABBBABABABBBABAABBABAAB"
  * string = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
  * string = hex representation: 88 76 a2 93 08 7d 7d 81 8a 58 12 22 8f 09 a3
==> passed
```

```
Test 17: check index() and length() with corner-case strings
  * a.txt
  * nomatch.txt
  * zebra.txt
  * alphanum.txt
==> passed

Test 18: check index() and length() with periodic strings
  * stars.txt
  * couscous.txt
==> passed

Test 19: check index() and length() with unary strings
  * length 10 string
  * length 100 string
  * length 1000 string
==> passed

Test 20: check index() and length() with random periodic strings
  * length 2 string over binary alphabet, repeated 2 times
  * length 3 string over binary alphabet, repeated 10 times
  * length 4 string over binary alphabet, repeated 4 times
  * length 5 string over binary alphabet, repeated 3 times
  * length 6 string over binary alphabet, repeated 2 times
  * length 7 string over uppercase alphabet, repeated 2 times
  * length 8 string over uppercase alphabet, repeated 3 times
  * length 9 string over uppercase alphabet, repeated 4 times
==> passed

Total: 20/20 tests passed!


================================================================
Testing correctness of MoveToFront
*-----------------------------------------------------------
Running 25 total tests.

Test 1a: check main() on text files
  * java MoveToFront - < abra.txt
  * java MoveToFront - < zebra.txt
  * java MoveToFront - < amendments.txt
  * java MoveToFront - < aesop.txt
==> passed

Test 1b: check main() on text files
  * java MoveToFront + < abra.txt.mtf
  * java MoveToFront + < zebra.txt.mtf
  * java MoveToFront + < amendments.txt.mtf
  * java MoveToFront + < aesop.txt.mtf
==> passed

Test 2a: check parsing of argument "-" in main() on text files
  * java MoveToFront - < abra.txt
  * java MoveToFront - < zebra.txt
  * java MoveToFront - < amendments.txt
  * java MoveToFront - < aesop.txt
==> passed

Test 2b: check parsing of argument "+" in main() on text files
  * java MoveToFront + < abra.txt.mtf
  * java MoveToFront + < zebra.txt.mtf
  * java MoveToFront + < amendments.txt.mtf
  * java MoveToFront + < aesop.txt.mtf
==> passed

Test 3a: check that main() is consistent with encode() on text files
  * abra.txt
  * zebra.txt
  * amendments.txt
  * aesop.txt
==> passed

Test 3b: check that main() is consistent with decode() on text files
```

```
  * abra.txt.mtf
  * zebra.txt.mtf
  * amendments.txt.mtf
  * aesop.txt.mtf
==> passed

Test 4a: check encode() on text files
  * abra.txt
  * zebra.txt
  * amendments.txt
  * aesop.txt
  * stars.txt
  * alphanum.txt
  * a.txt
==> passed

Test 4b: check encode() on binary files
  * us.gif
  * CS_bricks.jpg
  * rand10K.bin
==> passed

Test 4c: check encode() on random inputs
  * 10 random characters from { A } alphabet
  * 10 random characters from { A, B } alphabet
  * 10 random characters from { A, T, C, G } alphabet
  * 10 random characters from uppercase letter alphabet
  * 1000 random characters from { A } alphabet
  * 1000 random characters from { A, B } alphabet
  * 1000 random characters from { A, T, C, G } alphabet
  * 1000 random characters from uppercase letter alphabet
==> passed

Test 4d: check encode() on more random inputs
  * 1000 random characters from ASCII alphabet
  * 1000 random characters from extended ASCII alphabet
  * 1000 random characters from extended ASCII alphabet (excluding 0x00)
  * 1000 random characters from extended ASCII alphabet (excluding 0xFF)
==> passed

Test 5a: check decode() on move-to-front-encoded text files
  * abra.txt.mtf
  * zebra.txt.mtf
  * amendments.txt.mtf
  * aesop.txt.mtf
  * stars.txt.mtf
  * alphanum.txt.mtf
  * a.txt.mtf
==> passed

Test 5b: check decode() on move-to-front encoded binary files
  * us.gif.mtf
  * CS_bricks.jpg.mtf
  * rand10K.bin.mtf
==> passed

Test 5c: check decode() on random inputs
  * 10 random characters from { A } alphabet
  * 10 random characters from { A, B } alphabet
  * 10 random characters from { A, T, C, G } alphabet
  * 10 random characters from uppercase letter alphabet
  * 1000 random characters from { A } alphabet
  * 1000 random characters from { A, B } alphabet
  * 1000 random characters from { A, T, C, G } alphabet
  * 1000 random characters from uppercase letter alphabet
==> passed

Test 5d: check decode() on more random inputs
  * 1000 random characters from ASCII alphabet
  * 1000 random characters from extended ASCII alphabet
  * 1000 random characters from extended ASCII alphabet (excluding 0x00)
  * 1000 random characters from extended ASCII alphabet (excluding 0xFF)
==> passed
```

```
Test 5e: check decode() on random inputs
         that were encoded with move-to-front
  * 10 random characters from { A } alphabet
  * 10 random characters from { A, B } alphabet
  * 10 random characters from { A, T, C, G } alphabet
  * 10 random characters from uppercase letter alphabet
  * 1000 random characters from { A } alphabet
  * 1000 random characters from { A, B } alphabet
  * 1000 random characters from { A, T, C, G } alphabet
  * 1000 random characters from uppercase letter alphabet
==> passed

Test 5f: check decode() on more random inputs
         that were encoded with move-to-front
  * 1000 random characters from ASCII alphabet
  * 1000 random characters from extended ASCII alphabet
  * 1000 random characters from extended ASCII alphabet (excluding 0x00)
  * 1000 random characters from extended ASCII alphabet (excluding 0xFF)
==> passed

Test 6a: check whether decode(encode()) = original on text files
  * abra.txt
  * zebra.txt
  * amendments.txt
  * aesop.txt
  * stars.txt
  * alphanum.txt
  * a.txt
==> passed

Test 6b: check whether decode(encode()) = original on binary files
  * us.gif
  * CS_bricks.jpg
  * rand10K.bin
==> passed

Test 6c: check that decode(encode()) = original on random inputs
  * 10 random characters from { A } alphabet
  * 10 random characters from { A, B } alphabet
  * 10 random characters from { A, T, C, G } alphabet
  * 10 random characters from uppercase letter alphabet
  * 100 random characters from { A } alphabet
  * 1000 random characters from { A, B } alphabet
  * 1000 random characters from { A, T, C, G } alphabet
  * 1000 random characters from uppercase letter alphabet
==> passed

Test 6d: check that decode(encode()) = original on random inputs
  * 1000 random characters from ASCII alphabet
  * 1000 random characters from extended ASCII alphabet
  * 1000 random characters from extended ASCII alphabet (excluding 0x00)
  * 1000 random characters from extended ASCII alphabet (excluding 0xFF)
==> passed

Test 7a: check that encode() calls either close() or flush()
  * abra.txt
  * zebra.txt
  * amendments.txt
==> passed

Test 7b: check that decode() calls either close() or flush()
  * abra.txt.mtf
  * zebra.txt.mtf
  * amendments.txt.mtf
==> passed

Test 8a: check encode() on large files
  * aesop.txt
  * rand100K.bin
  * world192.txt
==> passed

Test 8b: check decode() on large files
  * aesop.txt.mtf
  * rand100K.bin.mtf
  * world192.txt.mtf
==> passed

Test 8c: check whether decode(encode()) = original on large files
  * aesop.txt
  * rand100K.bin
  * world192.txt
==> passed


Total: 25/25 tests passed!


================================================================
********************************************************************************
*  TESTING CORRECTNESS (substituting reference CircularSuffixArray)
********************************************************************************

Testing correctness of BurrowsWheeler
*-----------------------------------------------------------
Running 28 total tests.

Test 1a: check main() on text files
  * java BurrowsWheeler - < abra.txt
  * java BurrowsWheeler - < zebra.txt
  * java BurrowsWheeler - < cadabra.txt
  * java BurrowsWheeler - < amendments.txt
==> passed

Test 1b: check main() on text files
  * java BurrowsWheeler + < abra.txt.bwt
  * java BurrowsWheeler + < zebra.txt.bwt
  * java BurrowsWheeler + < cadabra.txt.bwt
  * java BurrowsWheeler + < amendments.txt.bwt
==> passed

Test 2a: check parsing of argument "-" in main() on text files
  * java BurrowsWheeler - < abra.txt
  * java BurrowsWheeler - < zebra.txt
  * java BurrowsWheeler - < cadabra.txt
  * java BurrowsWheeler - < amendments.txt
==> passed

Test 2b: check parsing of argument "+" in main() on text files
  * java BurrowsWheeler + < abra.txt.bwt
  * java BurrowsWheeler + < zebra.txt.bwt
  * java BurrowsWheeler + < cadabra.txt.bwt
  * java BurrowsWheeler + < amendments.txt.bwt
==> passed

Test 3a: check that main() is consistent with transform() on text files
  * abra.txt
  * zebra.txt
  * cadabra.txt
  * amendments.txt
==> passed

Test 3b: check that main() is consistent with inverseTransform() on text files
  * abra.txt.bwt
  * zebra.txt.bwt
  * cadabra.txt.bwt
  * amendments.txt.bwt
==> passed

Test 4a: check transform() on text files
  * abra.txt
  * zebra.txt
  * cadabra.txt
  * amendments.txt
==> passed

Test 4b: check transform() on corner-case text files
  * alphanum.txt
  * a.txt
==> passed

Test 4c: check transform() on binary files
  * us.gif
  * CS_bricks.jpg
  * rand10K.bin
==> passed

Test 4d: check transform() on random inputs
  * 10 random characters from binary alphabet
  * 10 random characters from DNA alphabet
  * 10 random characters from uppercase alphabet
  * 1000 random characters from binary alphabet
  * 1000 random characters from DNA alphabet
  * 1000 random characters from uppercase alphabet
==> passed

Test 4e: check transform() on more random inputs
  * 1000 random characters from ASCII alphabet
  * 1000 random characters from extended ASCII alphabet
  * 1000 random characters from extended ASCII alphabet (excluding 0x00)
  * 1000 random characters from extended ASCII alphabet (excluding 0xFF)
==> passed

Test 4f: check tranform() on random inputs that are circular
         shifts of themselves
  * 5 random strings from unary alphabet
  * 5 random strings from binary alphabet
  * 5 random strings from DNA alphabet
  * 5 random strings from uppercase alphabet
==> passed

Test 5a: check inverseTransform() on text files
  * abra.txt.bwt
  * zebra.txt.bwt
  * cadabra.txt.bwt
  * amendments.txt.bwt
==> passed

Test 5b: check inverseTransform() on corner-case text files
  * alphanum.txt.bwt
  * a.txt.bwt
  * stars.txt.bwt
  * couscous.txt.bwt
==> passed

Test 5c: check inverseTransform() on binary files
  * us.gif.bwt
  * CS_bricks.jpg.bwt
  * rand10K.bin.bwt
==> passed

Test 5d: check inverseTransform() of transform() on random inputs
  * 10 random characters from unary alphabet
  * 10 random characters from binary alphabet
  * 10 random characters from DNA alphabet
  * 10 random characters from uppercase alphabet
  * 100 random characters from unary alphabet
  * 1000 random characters from binary alphabet
  * 1000 random characters from DNA alphabet
  * 1000 random characters from uppercase alphabet
==> passed

Test 5e: check inverseTransform() of transform() on more random inputs
  * 1000 random characters from ASCII alphabet
  * 1000 random characters from extended ASCII alphabet
  * 1000 random characters from extended ASCII alphabet (excluding 0x00)
  * 1000 random characters from extended ASCII alphabet (excluding 0xFF)
==> passed

Test 6a: check that inverseTransform(transform()) = original on text files
  * abra.txt
  * zebra.txt
  * cadabra.txt
  * amendments.txt
==> passed

Test 6b: check that inverseTransform(transform()) = original on corner-case text files
  * alphanum.txt
  * a.txt
  * stars.txt
  * couscous.txt
==> passed

Test 6c: check that inverseTransform(transform()) = original on binary files
  * us.gif
  * CS_bricks.jpg
  * rand10K.bin
==> passed

Test 6d: check that inverseTransform(tranform()) = original on random inputs
  * 10 random characters from binary alphabet
  * 10 random characters from DNA alphabet
  * 10 random characters from uppercase alphabet
  * 1000 random characters from binary alphabet
  * 1000 random characters from DNA alphabet
  * 1000 random characters from uppercase alphabet
==> passed

Test 6e: check that inverseTransform(tranform()) = original on random inputs
  * 1000 random characters from ASCII alphabet
  * 1000 random characters from extended ASCII alphabet
  * 1000 random characters from extended ASCII alphabet (excluding 0x00)
  * 1000 random characters from extended ASCII alphabet (excluding 0xFF)
==> passed

Test 6f: check that inverseTransform(tranform()) = original
         on random inputs that are circular shifts of themselves
  * random strings from unary alphabet
  * random strings from binary alphabet
  * random strings from DNA alphabet
  * random strings from uppercase alphabet
==> passed

Test 7a: check that transform() calls either close() or flush()
  * abra.txt
  * zebra.txt
  * cadabra.txt
  * amendments.txt
==> passed

Test 7b: check that inverseTransform() calls either close() or flush()
  * abra.txt.bwt
  * zebra.txt.bwt
  * cadabra.txt.bwt
  * amendments.txt.bwt
==> passed

Test 8a: check transform() on large files
  * aesop.txt
  * rand100K.bin
  * world192.txt
==> passed

Test 8b: check inverseTransform() on large files
  * aesop.txt.bwt
  * rand100K.bin.bwt
  * world192.txt.bwt
==> passed

Test 8c: check that inverseTransform(transform()) = original on large files
  * aesop.txt
  * rand100K.bin
  * world192.txt
==> passed
```

```
Total: 28/28 tests passed!


================================================================
********************************************************************************
*  MEMORY
********************************************************************************

Analyzing memory of CircularSuffixArray
*-----------------------------------------------------------
Running 10 total tests.

Memory usage of a CircularSuffixArray for a random string of length n.
Maximum allowed memory is 64n + 128.

                 n        bytes
-------------------------------
=> passed       16          120
=> passed       32          184
=> passed       64          312
=> passed      128          568
=> passed      256         1080
=> passed      512         2104
=> passed     1024         4152
=> passed     2048         8248
=> passed     4096        16440
=> passed     8192        32824
==> 10/10 tests passed

Total: 10/10 tests passed!

Estimated student memory (bytes) = 4.00 n + 56.00   (R^2 = 1.000)

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



********************************************************************************
*  TIMING
********************************************************************************

Timing CircularSuffixArray
*-----------------------------------------------------------
Tests  1-13: time to create a circular suffix array for the firstRunning 26 total tests.


             n character of dickens.txt and call index(i) for each i

            [ max allowed time = 10 seconds and <= 12x reference ]

                 n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       9.76
=> passed     2000       0.00       0.00       1.22
=> passed     4000       0.00       0.00       1.21
=> passed     8000       0.00       0.00       1.32
=> passed    16000       0.01       0.00       1.39
=> passed    32000       0.01       0.01       1.65
=> passed    64000       0.02       0.01       1.90
=> passed   128000       0.04       0.02       1.88
=> passed   256000       0.08       0.03       2.65
=> passed   512000       0.13       0.06       2.31
=> passed  1024000       0.28       0.12       2.34
=> passed  2048000       0.59       0.26       2.27
=> passed  4096000       1.28       0.62       2.06

Estimated running time (using last 6 measurements)
    = 4.53e-07 * n^0.97  (R^2 = 0.99)


Tests 14-26: time to create circular suffix array for n random ASCII characters
            and call index(i) for each i

            [ max allowed time = 10 seconds and <= 20x reference ]
```

```
                 n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       3.15
=> passed     2000       0.00       0.00       1.92
=> passed     4000       0.00       0.00       1.86
=> passed     8000       0.00       0.00       3.11
=> passed    16000       0.00       0.00       4.57
=> passed    32000       0.01       0.00       5.84
=> passed    64000       0.01       0.00       5.81
=> passed   128000       0.03       0.01       5.02
=> passed   256000       0.06       0.02       3.46
=> passed   512000       0.12       0.03       3.37
=> passed  1024000       0.26       0.06       4.42
=> passed  2048000       0.55       0.10       5.24
=> passed  4096000       1.20       0.25       4.90

Estimated running time (using last 6 measurements)
    = 5.73e-08 * n^1.11  (R^2 = 1.00)


Total: 26/26 tests passed!


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



********************************************************************************
*  TIMING
********************************************************************************

Timing MoveToFront
*-----------------------------------------------------------
Running 40 total tests.

Test 1: count calls to methods in BinaryStdOut from encode()
  * abra.txt
  * amendments.txt
==> passed

Test 2: count calls to methods in BinaryStdOut from decode()
  * abra.txt.mtf
  * amendments.txt.mtf
==> passed

Test 3: count calls to methods in BinaryStdIn from encode()
  * abra.txt
  * amendments.txt
==> passed

Test 4: count calls to methods in BinaryStdIn from decode()
  * abra.txt.mtf
  * amendments.txt.mtf
==> passed

Tests  5-14: Timing encode() with first n character of dickens.txt
             [ max allowed time = 2 seconds and <= 4x reference ]

                 n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       1.16
=> passed     2000       0.00       0.00       1.73
=> passed     4000       0.01       0.00       1.73
=> passed     8000       0.01       0.01       1.75
=> passed    16000       0.03       0.01       1.77
=> passed    32000       0.05       0.03       1.79
=> passed    64000       0.10       0.06       1.79
=> passed   128000       0.21       0.12       1.79
=> passed   256000       0.41       0.23       1.79

Estimated running time (using last 6 measurements)
     = 1.85e-06 * n^0.99  (R^2 = 1.00)
```

```
Tests  15-22: Timing encode() with first n character of abab.txt
             [ max allowed time = 2 seconds and <= 4x reference ]

                  n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       1.93
=> passed     2000       0.00       0.00       1.91
=> passed     4000       0.01       0.00       1.91
=> passed     8000       0.01       0.01       1.89
=> passed    16000       0.03       0.01       1.88
=> passed    32000       0.05       0.03       1.87
=> passed    64000       0.10       0.05       1.88
=> passed   128000       0.20       0.11       1.88
=> passed   256000       0.40       0.22       1.87

Estimated running time (using last 6 measurements)
     = 1.62e-06 * n^1.00  (R^2 = 1.00)


Tests 23-31: Timing decode() with first n character of dickens.txt
             [ max allowed time = 2 seconds and <= 4x reference ]

                  n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       2.32
=> passed     2000       0.00       0.00       2.26
=> passed     4000       0.01       0.00       2.28
=> passed     8000       0.02       0.01       2.32
=> passed    16000       0.03       0.01       2.33
=> passed    32000       0.07       0.03       2.35
=> passed    64000       0.13       0.06       2.34
=> passed   128000       0.26       0.11       2.35
=> passed   256000       0.53       0.22       2.35

Estimated running time (using last 6 measurements)
     = 2.23e-06 * n^0.99  (R^2 = 1.00)


Tests 32-40: Timing decode() with first n character of abab.txt
             [ max allowed time = 2 seconds and <= 4x reference ]

                  n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       2.37
=> passed     2000       0.00       0.00       2.39
=> passed     4000       0.01       0.00       2.40
=> passed     8000       0.02       0.01       2.38
=> passed    16000       0.03       0.01       2.39
=> passed    32000       0.06       0.03       2.37
=> passed    64000       0.13       0.05       2.41
=> passed   128000       0.26       0.11       2.38
=> passed   256000       0.51       0.22       2.36

Estimated running time (using last 6 measurements)
     = 2.04e-06 * n^1.00  (R^2 = 1.00)


Total: 40/40 tests passed!


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




********************************************************************************
*  TIMING (substituting reference CircularSuffixArray)
********************************************************************************

Timing BurrowsWheeler
*-----------------------------------------------------------
Running 97 total tests.

Test 1: count calls to methods in CircularSuffixArray from transform()
```

```
  * abra.txt
  * amendments.txt
==> passed

Test 2: count calls to methods in CircularSuffixArray from inverseTransform()
  * abra.txt.bwt
  * amendments.txt.bwt
==> passed

Test 3: count calls to methods in BinaryStdOut from transform()
  * abra.txt
  * amendments.txt
==> passed

Test 4: count calls to methods in BinaryStdOut from inverseTransform()
  * abra.txt.bwt
  * amendments.txt.bwt
==> passed

Test 5: count calls to methods in BinaryStdIn from transform()
  * abra.txt
  * amendments.txt
==> passed

Test 6: count calls to methods in BinaryStdIn from inverseTransform()
  * abra.txt.bwt
  * amendments.txt.bwt
==> passed

Tests  7-19: timing transform() with first n character of dickens.txt
             [ max allowed time = 2 seconds and <= 8x reference ]

                 n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       0.12
=> passed     2000       0.00       0.00       0.45
=> passed     4000       0.00       0.00       0.61
=> passed     8000       0.00       0.00       0.55
=> passed    16000       0.00       0.01       0.42
=> passed    32000       0.00       0.01       0.55
=> passed    64000       0.01       0.01       0.63
=> passed   128000       0.02       0.03       0.69
=> passed   256000       0.04       0.06       0.66
=> passed   512000       0.07       0.09       0.77
=> passed  1024000       0.14       0.13       1.06
=> passed  2048000       0.30       0.29       1.04
=> passed  4096000       0.72       0.69       1.04

Estimated running time as a function of n (using last 6 measurements)
    = 1.07e-07 * n^1.02  (R^2 = 0.99)


Tests 20-32: timing transform() with first n character of random.bin
             [ max allowed time = 2 seconds and <= 8x reference ]

                 n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       1.00
=> passed     2000       0.00       0.00       1.07
=> passed     4000       0.00       0.00       1.01
=> passed     8000       0.00       0.00       1.04
=> passed    16000       0.00       0.00       1.09
=> passed    32000       0.00       0.00       0.99
=> passed    64000       0.01       0.01       1.09
=> passed   128000       0.02       0.01       1.07
=> passed   256000       0.03       0.03       1.04
=> passed   512000       0.07       0.07       1.05
=> passed  1024000       0.16       0.15       1.04
=> passed  2048000       0.34       0.33       1.05
=> passed  4096000       0.80       0.77       1.05

Estimated running time as a function of n (using last 6 measurements)
    = 2.47e-08 * n^1.13  (R^2 = 1.00)
```

```
Tests 33-45: timing transform() with first n character of abab.txt
             [ max allowed time = 2 seconds and <= 8x reference ]

                 n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       0.78
=> passed     2000       0.00       0.00       0.98
=> passed     4000       0.00       0.00       1.00
=> passed     8000       0.00       0.00       1.00
=> passed    16000       0.00       0.00       0.99
=> passed    32000       0.00       0.00       1.02
=> passed    64000       0.00       0.00       1.12
=> passed   128000       0.00       0.00       0.99
=> passed   256000       0.01       0.01       1.04
=> passed   512000       0.01       0.01       1.04
=> passed  1024000       0.03       0.02       1.02
=> passed  2048000       0.05       0.05       1.06
=> passed  4096000       0.10       0.10       1.03

Estimated running time as a function of n (using last 6 measurements)
    = 2.55e-08 * n^1.00  (R^2 = 1.00)


Tests 46-58: timing inverseTransform() with first n character of dickens.txt
             [ max allowed time = 2 seconds and <= 8x reference ]

                 n    student  reference      ratio
---------------------------------------------------
=> passed     1000       0.00       0.00       0.66
=> passed     2000       0.00       0.00       0.35
=> passed     4000       0.00       0.00       0.33
=> passed     8000       0.00       0.00       0.31
=> passed    16000       0.00       0.00       0.29
=> passed    32000       0.00       0.00       0.51
=> passed    64000       0.00       0.00       0.48
=> passed   128000       0.00       0.00       1.04
=> passed   256000       0.01       0.01       1.12
=> passed   512000       0.02       0.02       1.06
=> passed  1024000       0.04       0.04       1.03
=> passed  2048000       0.10       0.10       1.04
=> passed  4096000       0.30       0.25       1.19

Estimated running time as a function of n (using last 6 measurements)
    = 7.91e-10 * n^1.29  (R^2 = 1.00)


Tests 59-71: timing inverseTransform() with first n character of random.bin
             [ max allowed time = 2 seconds and <= 8x reference ]

                 n    student  reference      ratio
---------------------------------------------------
=> passed     1024       0.00       0.00       0.95
=> passed     2048       0.00       0.00       1.00
=> passed     4096       0.00       0.00       1.01
=> passed     8192       0.00       0.00       1.01
=> passed    16384       0.00       0.00       1.02
=> passed    32768       0.00       0.00       1.03
=> passed    65536       0.00       0.00       1.10
=> passed   131072       0.00       0.00       1.08
=> passed   262144       0.01       0.01       1.13
=> passed   524288       0.02       0.02       1.07
=> passed  1048576       0.04       0.04       1.04
=> passed  2097152       0.11       0.11       1.08
=> passed  4194304       0.35       0.31       1.12

Estimated running time as a function of n (using last 6 measurements)
    = 6.15e-10 * n^1.31  (R^2 = 1.00)


Tests 72-84: timing inverseTransform() with first n character of abab.txt
             [ max allowed time = 2 seconds and <= 8x reference ]

                 n    student  reference      ratio
```

```
---------------------------------------------------
=> passed     1000       0.00       0.00       0.95
=> passed     2000       0.00       0.00       1.00
=> passed     4000       0.00       0.00       0.97
=> passed     8000       0.00       0.00       1.03
=> passed    16000       0.00       0.00       0.98
=> passed    32000       0.00       0.00       1.03
=> passed    64000       0.00       0.00       1.03
=> passed   128000       0.00       0.00       1.03
=> passed   256000       0.01       0.01       1.04
=> passed   512000       0.01       0.01       1.05
=> passed  1024000       0.02       0.02       1.04
=> passed  2048000       0.04       0.04       1.04
=> passed  4096000       0.09       0.09       1.04

Estimated running time as a function of n (using last 6 measurements)
    = 2.06e-08 * n^1.00  (R^2 = 1.00)


Tests 85-97: timing inverseTransform() with first n character of cyclic.bin
             [ max allowed time = 2 seconds and <= 8x reference ]

                  n    student  reference      ratio
---------------------------------------------------
=> passed     1024       0.00       0.00       0.93
=> passed     2048       0.00       0.00       0.79
=> passed     4096       0.00       0.00       1.01
=> passed     8192       0.00       0.00       1.02
=> passed    16384       0.00       0.00       1.07
=> passed    32768       0.00       0.00       1.04
=> passed    65536       0.00       0.00       1.03
=> passed   131072       0.00       0.00       1.09
=> passed   262144       0.01       0.01       1.15
=> passed   524288       0.02       0.02       1.32
=> passed  1048576       0.04       0.04       0.93
=> passed  2097152       0.07       0.08       0.83
=> passed  4194304       0.18       0.12       1.49

Estimated running time as a function of n (using last 6 measurements)
    = 6.07e-09 * n^1.13  (R^2 = 0.99)


Total: 97/97 tests passed!


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