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

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     FAILED (1 warning)
PMD:          FAILED (1 warning)
Checkstyle:   FAILED (0 errors, 1 warning)

Correctness:  25/33 tests passed
Memory:       No tests available for autograding.
Timing:       No tests available for autograding.

Aggregate score: 78.18%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 90% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
 579 May  2 21:42 Checkerboard.java
 815 May  2 21:42 ShannonEntropy.java
1.2K May  2 21:42 WorldMap.java


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


% javac ShannonEntropy.java
*-----------------------------------------------------------

% javac Checkerboard.java
*-----------------------------------------------------------

% javac WorldMap.java
*-----------------------------------------------------------


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


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

Checkerboard:

WorldMap:

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


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


% spotbugs *.class
*-----------------------------------------------------------
M C FPL_FLOATING_POINT_LOOPS FPL: Method Checkerboard.main(String[]) uses floating point indexed loops  At Checkerboard.java:[line 5]
SpotBugs ends with 1 warning.


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


% pmd .
*-----------------------------------------------------------
WorldMap.java:15: Avoid unused local variables, such as 'name'. [UnusedLocalVariable]
PMD ends with 1 warning.


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


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

% custom checkstyle checks for ShannonEntropy.java
*-----------------------------------------------------------
[WARN] ShannonEntropy.java:1: The number (0) of calls to 'Integer.parseInt()' must equal the number (1) of integer command-line arguments. [C
Checkstyle ends with 0 errors and 1 warning.
```

```
% custom checkstyle checks for Checkerboard.java
*-----------------------------------------------------------

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


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


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

Testing correctness of ShannonEntropy
*-----------------------------------------------------------
Running 12 total tests.

Test 1: check output format
  % java-introcs ShannonEntropy 2 < fair-coin.txt
  0.9980

  % java-introcs ShannonEntropy 6 < loaded-die.txt
  1.8566

  % java-introcs ShannonEntropy 100 < letters100.txt
  4.3788

==> passed

Test 2: check that program reads all data from standard input
  * java-introcs ShannonEntropy < fair-coin.txt
  * java-introcs ShannonEntropy < loaded-die.txt
  * java-introcs ShannonEntropy < letters100.txt
==> passed

Test 3: check correctness for fixed inputs
  * java-introcs ShannonEntropy 2 < fair-coin.txt
    - student   entropy = 0.998
    - reference entropy = 1.0
    - input file:
      1 1 1 1 2 1 2 1 1 2
      2 2 2 2 1 2 1 2 2 1

  * java-introcs ShannonEntropy 2 < biased-coin.txt
    - student   entropy = 0.7425
    - reference entropy = 0.7219
    - input file:
      1 1 1 1 1 1 1 1 1 2
      2 1 1 1 1 2 1 2 1 1

  * java-introcs ShannonEntropy 6 < loaded-die.txt
    - student   entropy = 1.8566
    - reference entropy = 1.875
    - input file:
      3 2 6 2 4 3 2 1 2 2 1 3 2 3 2 2

  * java-introcs ShannonEntropy 100 < letters100.txt
    - student   entropy = 4.3788
    - reference entropy = 4.3798

  * java-introcs ShannonEntropy 3 < roshambo.txt
    - student   entropy = 0.2404
    - reference entropy = 1.585
    - input file:
      1 2 3 1 2 3 1 2 3
      1 1 1 2 2 2 3 3 3

==> FAILED

Test 4: check correctness when entropy is 0
  * m = 1, all 1s
  * m = 2, all 1s
  * m = 2, all 2s

    java.lang.ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 2

    ShannonEntropy.main(ShannonEntropy.java:10)
    UtilCOS.execute(UtilCOS.java:93)
    UtilCOS.execute(UtilCOS.java:76)
    TestShannonEntropy.checkCorrectnessEntropyZero(TestShannonEntropy.java:163)
    TestShannonEntropy.test4(TestShannonEntropy.java:247)
    TestShannonEntropy.main(TestShannonEntropy.java:379)

  * m = 6, all 1s
  * m = 6, all 3s

    java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 2

    ShannonEntropy.main(ShannonEntropy.java:10)
    UtilCOS.execute(UtilCOS.java:93)
    UtilCOS.execute(UtilCOS.java:76)
    TestShannonEntropy.checkCorrectnessEntropyZero(TestShannonEntropy.java:163)
    TestShannonEntropy.test4(TestShannonEntropy.java:249)
    TestShannonEntropy.main(TestShannonEntropy.java:379)

  * m = 6, all 6s
```

```
      java.lang.ArrayIndexOutOfBoundsException: Index 6 out of bounds for length 2

      ShannonEntropy.main(ShannonEntropy.java:10)
      UtilCOS.execute(UtilCOS.java:93)
      UtilCOS.execute(UtilCOS.java:76)
      TestShannonEntropy.checkCorrectnessEntropyZero(TestShannonEntropy.java:163)
      TestShannonEntropy.test4(TestShannonEntropy.java:250)
      TestShannonEntropy.main(TestShannonEntropy.java:379)

    * m = 100, all 100s

      java.lang.ArrayIndexOutOfBoundsException: Index 100 out of bounds for length 2

      ShannonEntropy.main(ShannonEntropy.java:10)
      UtilCOS.execute(UtilCOS.java:93)
      UtilCOS.execute(UtilCOS.java:76)
      TestShannonEntropy.checkCorrectnessEntropyZero(TestShannonEntropy.java:163)
      TestShannonEntropy.test4(TestShannonEntropy.java:251)
      TestShannonEntropy.main(TestShannonEntropy.java:379)

  ==> FAILED

  Test 5: check correctness for random inputs with p_i > 0 for all i
    * 10 random integers between 1 and 2
      - student   entropy = 0.9183
      - reference entropy = 0.971
      - failed on trial 1 of 100
      - m =  2
      - input file:
         1  1  2  2  2  2  2  2  1  1

    * 100 random integers between 1 and 2
      - student   entropy = 0.3298
      - reference entropy = 0.3274
      - failed on trial 1 of 100
      - m =  2

    * 20 random integers between 1 and 6
      - student   entropy = 2.524
      - reference entropy = 2.5037
      - failed on trial 1 of 100
      - m =  6
      - input file:
         4  2  4  5  2  6  6  3  5  5
         1  1  4  3  4  1  4  6  6  5

    * 1000 random integers between 1 and 6
      - student   entropy = 0.0419
      - reference entropy = 1.9389
      - failed on trial 1 of 10
      - m =  6

    * 1000 random integers between 1 and 26
      - student   entropy = 4.1466
      - reference entropy = 4.1458
      - failed on trial 1 of 10
      - m =  26

  ==> FAILED

  Test 6: check correctness for random inputs with p_m = 0
    * 10 random integers between 1 and 6
      - student   entropy = 1.1866
      - reference entropy = 2.171
      - failed on trial 1 of 100
      - m =  6
      - input file:
         1  1  1  5  4  3  5  4  2  4

    * 50 random integers between 1 and 10
      - student   entropy = 3.0584
      - reference entropy = 3.0665
      - failed on trial 1 of 100
      - m =  10

  ==> FAILED

  Test 7: check correctness for random inputs with p_i = 0 for some i
    * 10 random integers between 1 and 6
      - student   entropy = 1.3516
      - reference entropy = 1.4855
      - failed on trial 1 of 100
      - m =  6
      - input file:
         3  4  2  2  4  4  2  2  3  2

    * 100 random integers between 1 and 6
      - student   entropy = 1.11
      - reference entropy = 1.1174
      - failed on trial 1 of 100
      - m =  6

    * 50 random integers between 1 and 26
      - student   entropy = 3.7052
      - reference entropy = 3.7077
      - failed on trial 1 of 100
      - m =  26
    * 1000 random integers between 1 and 26
      - student   entropy = 4.0905
      - reference entropy = 4.0913
      - failed on trial 1 of 10
      - m =  26

  ==> FAILED

  Test 8: check correctness for random inputs with m = n
    * 2 random integers between 1 and 2

      java.lang.ArrayIndexOutOfBoundsException: Index 2 out of bounds for length 2

      ShannonEntropy.main(ShannonEntropy.java:10)
      UtilCOS.execute(UtilCOS.java:93)
      UtilCOS.execute(UtilCOS.java:76)
      TestShannonEntropy.checkCorrectnessRandom(TestShannonEntropy.java:134)
      TestShannonEntropy.test8(TestShannonEntropy.java:320)
      TestShannonEntropy.main(TestShannonEntropy.java:387)

    * 3 random integers between 1 and 3
      - student   entropy = 0.5
      - reference entropy = 1.585
      - failed on trial 1 of 100
      - m =  3
      - input file:
         1  2  3

    * 4 random integers between 1 and 4

      java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4

      ShannonEntropy.main(ShannonEntropy.java:10)
      UtilCOS.execute(UtilCOS.java:93)
      UtilCOS.execute(UtilCOS.java:76)
      TestShannonEntropy.checkCorrectnessRandom(TestShannonEntropy.java:134)
      TestShannonEntropy.test8(TestShannonEntropy.java:322)
      TestShannonEntropy.main(TestShannonEntropy.java:387)

    * 6 random integers between 1 and 6
      - student   entropy = 0.9288
      - reference entropy = 2.585
      - failed on trial 1 of 100
      - m =  6
      - input file:
         1  5  3  2  6  4

  ==> FAILED

  Test 9: check correctness for random inputs with m > n
    * 3 random integers between 1 and 6
      - student   entropy = 1.0
      - reference entropy = 0.9183
      - failed on trial 1 of 100
      - m =  6
      - input file:
         2  2  1

    * 10 random integers between 1 and 26

      java.lang.ArrayIndexOutOfBoundsException: Index 14 out of bounds for length 10

      ShannonEntropy.main(ShannonEntropy.java:10)
      UtilCOS.execute(UtilCOS.java:93)
      UtilCOS.execute(UtilCOS.java:76)
      TestShannonEntropy.checkCorrectnessRandom(TestShannonEntropy.java:134)
      TestShannonEntropy.test9(TestShannonEntropy.java:339)
      TestShannonEntropy.main(TestShannonEntropy.java:389)

  ==> FAILED

  Test 10: check correctness for inputs with large n
    * java-introcs ShannonEntropy 26 < letters.txt
  ==> passed

  Test 11: check correctness for inputs with large m
    * java-introcs ShannonEntropy 1000000 < ids20.txt

      java.lang.ArrayIndexOutOfBoundsException: Index 49561 out of bounds for length 10

      ShannonEntropy.main(ShannonEntropy.java:10)
      UtilCOS.execute(UtilCOS.java:93)
      UtilCOS.execute(UtilCOS.java:76)
      TestShannonEntropy.isCorrect(TestShannonEntropy.java:109)
      TestShannonEntropy.checkCorrectness(TestShannonEntropy.java:67)
      TestShannonEntropy.test11(TestShannonEntropy.java:352)
      TestShannonEntropy.main(TestShannonEntropy.java:393)

  ==> FAILED

  Test 12: check correctness for inputs with large m and n
    * java-introcs ShannonEntropy 1000000 < ids100000.txt
    * java-introcs ShannonEntropy 1000000 < ids1000000.txt
  ==> passed

  Total: 4/12 tests passed!
```

```
================================================================
Testing correctness of Checkerboard
*-----------------------------------------------------------
Running 12 total tests.

Test 1: check output format
  % java Checkerboard 1
  [no output]

  % java Checkerboard 2
  [no output]

  % java Checkerboard 3
  [no output]

  % java Checkerboard 4
  [no output]

  % java Checkerboard 5
  [no output]

  % java Checkerboard 6
  [no output]

==> passed

Test 2: check formatting of standard drawing
  * java Checkerboard 1
  [no output]

  * java Checkerboard 2
  [no output]

  * java Checkerboard 3
  [no output]

  * java Checkerboard 4
  [no output]

  * java Checkerboard 5
  [no output]

  * java Checkerboard 6
  [no output]

==> passed

Test 3: check number of filled polygons
  * n = 1
  * n = 2
  * n = 3
  * n = 4
  * n = 5
  * n = 6
==> passed

Test 4: check correctness, ignoring color
  * n = 1
  * n = 2
  * n = 3
  * n = 4
  * n = 5
  * n = 6
==> passed

Test 5: check that lower-left square is blue for odd n
  * n = 1
  * n = 3
  * n = 5
  * n = 7
  * n = 9
==> passed

Test 6: check that lower-left square is blue for even n
  * n = 2
  * n = 4
  * n = 6
  * n = 8
  * n = 10
==> passed

Test 7: check that lower-left square is blue for large n
  * 10 <= n < 20
  * 20 <= n < 30
  * 30 <= n < 50
==> passed

Test 8: check that it draws a checkerboard pattern for even n
  * n = 2
  * n = 4
  * n = 6
  * n = 8
  * n = 10
==> passed

Test 9: check that it draws a checkerboard pattern for odd n
  * n = 1
  * n = 3
```

```
    * n = 5
    * n = 7
    * n = 9
  ==> passed

  Test 10: check that it draws a checkerboard pattern for larger n
    * 10 <= n < 20
    * 20 <= n < 30
    * 30 <= n < 50
  ==> passed

  Test 11: check number of calls to StdDraw.setXscale() and StdDraw.setYscale()
    * n = 2
    * n = 4
    * n = 6
    * n = 8
    * n = 9
  ==> passed

  Test 12: check arguments to StdDraw.setXscale() and StdDraw.setYscale()
    * n = 2
    * n = 4
    * n = 6
    * n = 8
    * n = 9
  ==> passed


  Checkerboard Total: 12/12 tests passed!


  ================================================================
  Testing correctness of WorldMap
  *-----------------------------------------------------------
  Running 9 total tests.

  Test 1: check formatting of standard output
    % java WorldMap < square.txt
    [no output]

    % java WorldMap < shapes.txt
    [no output]

    % java WorldMap < usa.txt
    [no output]

    % java WorldMap < russia.txt
    [no output]

    % java WorldMap < india.txt
    [no output]

    % java WorldMap < world-low-resolution.txt
    [no output]

  ==> passed

  Test 2: check that program reads all data from standard input
    * java-introcs WorldMap < square.txt
    * java-introcs WorldMap < shapes.txt
    * java-introcs WorldMap < usa.txt
    * java-introcs WorldMap < russia.txt
    * java-introcs WorldMap < india.txt
  ==> passed

  Test 3: check formatting of standard drawing
    * java WorldMap < square.txt
    [no output]

    * java WorldMap < shapes.txt
    [no output]

    * java WorldMap < usa.txt
    [no output]

    * java WorldMap < russia.txt
    [no output]

    * java WorldMap < india.txt
    [no output]

    * java WorldMap < world-low-resolution.txt
    [no output]

  ==> passed

  Test 4: check number of polygons
    * file = square.txt
    * file = shapes.txt
    * file = usa.txt
    * file = russia.txt
    * file = india.txt
    * file = world-low-resolution.txt
  ==> passed

  Test 5: check correctness for input files
    * Square                         square.txt
    * Triange, Square, and Pentagon  shapes.txt
    * USA                          usa.txt
    * Russia                       russia.txt
    * India                        india.txt
    * World (low resolution)       world-low-resolution.txt
  ==> passed

  Test 6: check correctness for input files with one polygon
    * Hilbert polygon of order 1     hilbert1.txt
    * Hilbert polygon of order 2     hilbert2.txt
    * Hilbert polygon of order 3     hilbert3.txt
    * Hilbert polygon of order 4     hilbert4.txt
    * Hilbert polygon of order 5     hilbert5.txt
    * Hilbert polygon of order 6     hilbert6.txt
  ==> passed

  Test 7: check correctness for random countries
    * Andorra                      world/andorra.txt
    * Luxembourg                   world/luxembourg.txt
    * Oman                         world/oman.txt
    * Congo Dr                     world/congo-dr.txt
    * Estonia                      world/estonia.txt
    * China                        world/china.txt
    * Kazakhstan                   world/kazakhstan.txt
    * Norway                       world/norway.txt
    * Armenia                      world/armenia.txt
    * Cape Verde                   world/cape-verde.txt
    * Georgia                      world/georgia.txt
    * Tajikistan                   world/tajikistan.txt
    * Panama                       world/panama.txt
    * Egypt                        world/egypt.txt
    * Zambia                       world/zambia.txt
  ==> passed

  Test 8: check correctness for random states in USA
    * Louisiana                    usa/usa-la.txt
    * Maryland                     usa/usa-md.txt
    * Georgia                      usa/usa-ga.txt
    * New Jersey                   usa/usa-nj.txt
    * Maine                        usa/usa-me.txt
    * North Carolina               usa/usa-nc.txt
    * California                   usa/usa-ca.txt
    * Montana                      usa/usa-mt.txt
    * Utah                         usa/usa-ut.txt
    * South Dakota                 usa/usa-sd.txt
    * Pennsylvania                 usa/usa-pa.txt
    * Alaska                       usa/usa-ak.txt
    * Virginia                     usa/usa-va.txt
    * Rhode Island                 usa/usa-ri.txt
    * Colorado                     usa/usa-co.txt
  ==> passed

  Test 9: check correctness of large input files
    * USA (all counties)           usa-all-counties.txt
    * World                        world.txt
  ==> passed


  WorldMap Total: 9/9 tests passed!


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