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

ASSESSMENT SUMMARY

```
Compilation:  PASSED
API:          PASSED

SpotBugs:     PASSED
PMD:          PASSED
Checkstyle:   FAILED (0 errors, 3 warnings)

Correctness:  27/28 tests passed
Memory:       No tests available for autograding.
Timing:       No tests available for autograding.

Aggregate score: 96.79%
[ Compilation: 5%, API: 5%, Style: 0%, Correctness: 90% ]
```

ASSESSMENT DETAILS

```
The following files were submitted:
----------------------------------
 958 Aug 21 02:20 Birthday.java
 723 Aug 21 02:20 DiscreteDistribution.java
1.5K Aug 21 02:20 Minesweeper.java
 769 Aug 21 02:20 ThueMorse.java


================================================================================
*  COMPILING
================================================================================


% javac DiscreteDistribution.java
*------------------------------------------------------------

% javac ThueMorse.java
*------------------------------------------------------------

% javac Birthday.java
*------------------------------------------------------------

% javac Minesweeper.java
*------------------------------------------------------------


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

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

ThueMorse:

Birthday:

Minesweeper:

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


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


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


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


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


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


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

```
[WARN] DiscreteDistribution.java:8:15: The local variable 'S' must start with a lowercase letter and use camelCase. [LocalVariableName]
Checkstyle ends with 0 errors and 1 warning.

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

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

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

% custom checkstyle checks for Minesweeper.java
*-----------------------------------------------------------
[WARN] Minesweeper.java:15: Calling 'Math.random()' in more than one place suggests poor design in this program. [Design]
[WARN] Minesweeper.java:31:37: The numeric literal '8' appears to be unnecessary. [NumericLiteral]
Checkstyle ends with 0 errors and 2 warnings.


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


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

Testing correctness of DiscreteDistribution
*-----------------------------------------------------------
Running 6 total tests.

Test 1: check output format
  % java DiscreteDistribution 9 1 1 1 1 1 1
  3 3 3 5 5 2 4 5 3

  % java DiscreteDistribution 8 10 20 30 40 50 60 50 40 30 20 10
  11 8 6 5 11 6 6 5

  % java DiscreteDistribution 7 10 10 10 10 10 50
  2 3 6 4 5 6 2

  % java DiscreteDistribution 6 50 50
  1 2 1 2 1 1

  % java DiscreteDistribution 5 80 20
  1 2 1 1 1

  % java DiscreteDistribution 4 301 176 125 97 79 67 58 51 46
  1 4 3 3

  % java DiscreteDistribution 3 19 49 60 47 32 18 3 3 1
  1 6 2

  % java DiscreteDistribution 2 9316001 10274874 10109130 10045436 9850199 6704495 5886889
  1 5

  % java DiscreteDistribution 1 8167 1492 2782 4253 12702 2228 2015 6094 6966 153 772     ...
  12

==> passed

Test 2: check that output contains correct number of integers
  * fair die                          [ repeated 1000 times ]
  * sum of two dice                   [ repeated 1000 times ]
  * loaded die                        [ repeated 1000 times ]
  * fair coin                         [ repeated 1000 times ]
  * 80/20 biased coin                 [ repeated 1000 times ]
  * 9 digits in Benford's law         [ repeated 1000 times ]
  * goals in FIFA World Cup 1990-2002 [ repeated 1000 times ]
  * U.S. birthdays by day of week     [ repeated 1000 times ]
  * 26 letters in English language    [ repeated 1000 times ]
==> passed

Test 3: check that output is a sequence of integers between 1 and n
  * fair die                          [ repeated 1000 times ]
  * sum of two dice                   [ repeated 1000 times ]
  * loaded die                        [ repeated 1000 times ]
  * fair coin                         [ repeated 1000 times ]
  * 80/20 biased coin                 [ repeated 1000 times ]
  * 9 digits in Benford's law         [ repeated 1000 times ]
  * goals in FIFA World Cup 1990-2002 [ repeated 1000 times ]
  * U.S. birthdays by day of week     [ repeated 1000 times ]
  * 26 letters in English language    [ repeated 1000 times ]
==> passed

Test 4: check that program produces different results when run twice
  * fair die                          [ repeated 10 times ]
  * sum of two dice                   [ repeated 12 times ]
  * loaded die                        [ repeated 10 times ]
  * fair coin                         [ repeated 20 times ]
  * 80/20 biased coin                 [ repeated 30 times ]
  * 9 digits in Benford's law         [ repeated 10 times ]
  * goals in FIFA World Cup 1990-2002 [ repeated 10 times ]
  * U.S. birthdays by day of week     [ repeated 14 times ]
  * 26 letters in English language    [ repeated 10 times ]
```

```
==> passed

Test 5: check randomness
  * fair die                           [ repeated 100000 times ]
  * sum of two dice                    [ repeated 100000 times ]
  * loaded die                         [ repeated 100000 times ]
  * fair coin                          [ repeated 100000 times ]
  * 80/20 biased coin                  [ repeated 100000 times ]
  * 9 digits in Benford's law          [ repeated 100000 times ]
  * 26 letters in English language     [ repeated 100000 times ]
  * goals in FIFA World Cup 1990-2002  [ repeated 100000 times ]
  * U.S. birthdays by day of week      [ repeated 100000 times ]
==> passed

Test 6: check randomness when n = 1
  * a_1 = 1                            [ repeated 100000 times ]
  * a_1 = 100                          [ repeated 100000 times ]
==> passed


DiscreteDistribution Total: 6/6 tests passed!


================================================================
Testing correctness of ThueMorse
*----------------------------------------------------------
Running 5 total tests.

Test 1: check output format
  % java ThueMorse 2
  +  -
  -  +

  % java ThueMorse 4
  +  -  -  +
  -  +  +  -
  -  +  +  -
  +  -  -  +

  % java ThueMorse 8
  +  -  -  +  -  +  +  -
  -  +  +  -  +  -  -  +
  -  +  +  -  +  -  -  +
  +  -  -  +  -  +  +  -
  -  +  +  -  +  -  -  +
  +  -  -  +  -  +  +  -
  +  -  -  +  -  +  +  -
  -  +  +  -  +  -  -  +

  % java ThueMorse 16
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  -  +  +  -  +  -  -  +  +  -  -  +  -  +  +  -
  +  -  -  +  -  +  +  -  -  +  +  -  +  -  -  +

==> passed

Test 2: check correctness when n is a power of 2
  * java ThueMorse 2
  * java ThueMorse 4
  * java ThueMorse 8
  * java ThueMorse 16
  * java ThueMorse 32
  * java ThueMorse 64
==> passed

Test 3: check correctness when n is not a power of 2
  * java ThueMorse 3
  * java ThueMorse 5
  * java ThueMorse 6
  * java ThueMorse 7
  * java ThueMorse 9
  * java ThueMorse 10
  * java ThueMorse 11
  * java ThueMorse 12
  * java ThueMorse 13
  * java ThueMorse 14
  * java ThueMorse 15
==> passed

Test 4: check corner case
  * java ThueMorse 1
```

```
==> passed

Test 5: check random values of n
  * 100 random values of n in [16, 32)
  * 100 random values of n in [32, 64)
  * 50 random values of n in [64, 128)
  * 25 random values of n in [128, 256)
==> passed


ThueMorse Total: 5/5 tests passed!


================================================================
Testing correctness of Birthday
*----------------------------------------------------------
Running 6 total tests.

Test 1: check output format
  % java Birthday 365 100000
  1     0       0.00000
  2     288     0.00288
  3     587     0.00875
  4     813     0.01688
  5     1090    0.02778
  6     1367    0.04145
  7     1634    0.05779
  8     1874    0.07653
  9     2020    0.09673
  10    2240    0.11913
  11    2378    0.14291
  12    2566    0.16857
  13    2755    0.19612
  14    2878    0.22490
  15    2895    0.25385
  16    3052    0.28437
  17    3179    0.31616
  18    3186    0.34802
  19    3112    0.37914
  20    3179    0.41093
  21    3179    0.44272
  22    3157    0.47429
  23    3141    0.50570

  % java Birthday 31 100000
  1     0       0.00000
  2     3241    0.03241
  3     6189    0.09430
  4     8783    0.18213
  5     10569   0.28782
  6     11534   0.40316
  7     11514   0.51830

==> passed

Test 2: check values in first column
  * java Birthday 365 10000
  * java Birthday 31 10000
  * java Birthday 1 1000
  * java Birthday 2 1000
==> passed

Test 3: check that cumulative percentages are monotone nondecreasing
        and table stops when percentage reaches (or exceeds) 50%
  * java Birthday 365 10000 [ repeated 10 times ]
  * java Birthday 31 10000 [ repeated 10 times ]
  * java Birthday 10 5 [ repeated 1000 times ]
  * java Birthday 4 4 [ repeated 1000 times ]
    - cumulative percentage must reach (or exceed) 50%
    - last student cumulative percentage = 0.25
    - student output:
      1 0       0.00000
      2 1       0.25000
      3 0       0.25000
      4 0       0.25000

    - failed on trial 871 of 1000

  * java Birthday 2 2 [ repeated 1000 times ]
    - cumulative percentage must reach (or exceed) 50%
    - last student cumulative percentage = 0.0
    - student output:
      1 0       0.00000
      2 0       0.00000

    - failed on trial 2 of 1000

==> FAILED

Test 4: check that cumulative percentages are consistent with frequencies
  * java Birthday 365 10000
  * java Birthday 31 10000
==> passed
```

```
Test 5: check that each execution of program outputs a different table
  * java Birthday 365 10000 [ repeated twice ]
  * java Birthday 31 10000 [ repeated twice ]
==> passed

Test 6: check randomness of birthdays
  * java Birthday 365 1000000
  * java Birthday 31 1000000
  * java Birthday 7 1000000
  * java Birthday 5 1000000
==> passed


Birthday Total: 5/6 tests passed!


================================================================
Testing correctness of Minesweeper
*----------------------------------------------------------
Running 11 total tests.

Test 1: check output format
  % java Minesweeper 9 9 10
  1  2  4  *  2  0  0  0  0
  2  *  *  *  2  1  1  1  0
  2  *  4  3  2  2  *  1  0
  1  1  1  1  *  2  1  1  0
  0  0  0  1  1  1  0  0  0
  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  1  1
  1  1  1  1  1  0  0  1  *
  *  1  1  *  1  0  0  1  1

  % java Minesweeper 16 16 40
  0  0  0  0  1  1  1  0  0  0  0  1  *  1  1  *
  1  1  0  0  1  *  1  0  0  0  0  1  1  1  1  1
  *  2  1  1  3  2  3  1  1  1  1  1  1  1  1  0
  *  3  1  *  3  *  2  *  1  1  *  1  1  *  1  0
  *  3  1  2  *  2  2  1  1  1  1  1  1  2  1  1
  *  4  2  2  1  1  0  0  0  1  1  1  0  1  2  *
  2  *  *  1  1  1  1  0  0  1  *  1  0  1  *  2
  1  2  2  2  3  *  4  2  1  1  1  1  0  1  1  1
  1  1  1  1  *  *  *  *  1  0  1  1  1  1  1  1
  1  *  2  2  3  3  3  2  2  1  2  *  1  1  *  2
  1  1  2  *  2  1  1  0  1  *  2  1  1  1  2  *
  0  0  1  2  3  *  1  0  1  1  1  0  0  0  1  1
  0  1  2  4  *  3  1  0  0  0  1  1  1  0  0  0
  0  1  *  *  *  2  1  2  2  1  2  *  3  1  0  0
  1  2  3  3  2  1  1  *  *  1  2  *  *  1  0  0
  1  *  1  0  0  0  1  2  2  1  1  2  2  1  0  0

  % java Minesweeper 16 30 82
  1  1  1  0  0  0  0  0  0  0  1  *  2  *  1  0  0  0  0  0  0  2  *  2  0  0  0  1  *
  1  *  2  1  1  1  1  1  0  0  1  1  2  1  2  1  1  0  0  1  1  2  4  *  3  0  0  0  1  1
  1  1  3  *  2  1  *  2  2  1  1  1  1  1  2  *  3  1  1  2  *  3  *  *  3  1  1  1  0  0
  0  0  2  *  4  3  3  *  2  *  1  1  *  1  2  *  4  *  1  2  *  3  3  *  3  2  *  1  1  1
  0  0  1  2  *  *  3  2  2  2  2  2  1  1  1  3  *  3  2  2  3  2  2  1  2  *  3  2  1  *
  0  0  0  1  2  3  *  1  0  1  *  1  0  1  1  3  *  2  2  *  4  *  2  0  1  2  *  2  2  1
  0  1  1  1  1  2  2  1  0  1  1  1  0  1  *  2  2  2  3  *  4  *  2  1  1  2  2  *  2  1
  0  1  *  1  2  *  2  0  0  0  0  0  0  1  1  1  1  *  3  2  3  1  1  1  *  1  1  1  2  *
  1  2  2  2  3  *  2  0  0  0  0  0  1  1  1  0  2  2  3  *  1  0  0  1  1  1  0  0  1  1
  1  *  2  2  *  3  2  1  1  1  1  1  3  *  2  0  1  *  2  1  1  0  0  1  1  1  0  0  0  0
  1  1  2  *  2  3  *  2  1  *  1  1  *  *  3  0  1  1  2  1  1  0  0  1  *  1  0  0  0  0
  0  0  1  2  3  5  *  4  3  2  2  1  3  *  3  1  0  0  1  *  2  1  2  2  2  1  1  1  2  1
  0  0  1  2  *  *  *  *  3  *  2  0  2  3  *  1  0  0  1  1  3  *  3  *  1  0  1  *  3  *
  1  1  2  *  5  *  4  2  4  *  4  1  1  *  2  1  0  0  0  0  3  *  6  4  3  1  1  1  3  *
  1  *  2  2  *  2  1  0  2  *  *  2  2  2  2  1  0  0  0  0  2  *  *  *  *  1  0  0  1  1
  1  1  1  1  1  1  0  0  1  2  3  *  1  1  *  1  0  0  0  0  1  2  3  3  2  1  0  0  0  0

  % java Minesweeper 4 8 0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0

  % java Minesweeper 8 4 32
  *  *  *  *
  *  *  *  *
  *  *  *  *
  *  *  *  *
  *  *  *  *
  *  *  *  *
  *  *  *  *
  *  *  *  *

  % java Minesweeper 1 20 10
  *  1  1  *  2  *  *  *  *  2  *  1  0  1  *  2  *  1  1  *

==> passed

Test 2: check that counts are consistent with mines (varying k)
  * m =  4, n =  8, k random [1000 trials]
  * m =  8, n =  4, k random [1000 trials]
  * m =  5, n = 40, k random [1000 trials]
```

```
  * m =  7, n = 30, k random [1000 trials]
  * m = 10, n = 10, k random [1000 trials]
==> passed

Test 3: check that counts are consistent with mines (fixed k)
  * k =  1, m and n random   [1000 trials]
  * k = 10, m and n random   [1000 trials]
  * k = 20, m and n random   [1000 trials]
  * k = 50, m and n random   [1000 trials]
  * k = 80, m and n random   [1000 trials]
  * k = 90, m and n random   [1000 trials]
  * k = 99, m and n random   [1000 trials]
==> passed

Test 4: check that counts are consistent with mines (corner cases)
  * m =  5, n = 10, k =  0
  * m = 10, n =  5, k =  0
  * m =  5, n = 10, k = 50
  * m = 10, n =  5, k = 50
  * k =  0, m and n random   [1000 trials]
  * k =  1, m and n random   [1000 trials]
==> passed

Test 5: check that program produces different results each time
  * m =  4, n =  8, k = 16 [2 trials]
  * m =  8, n =  4, k = 26 [2 trials]
  * m =  1, n = 20, k = 16 [2 trials]
  * m = 20, n =  1, k = 10 [2 trials]
==> passed

Test 6: check number of mines, with k varying
  * m =  4, n =  8, k random [1000 trials]
  * m =  8, n =  4, k random [1000 trials]
  * m =  5, n = 40, k random [1000 trials]
  * m =  7, n = 30, k random [1000 trials]
  * m = 10, n = 10, k random [1000 trials]
==> passed

Test 7: check number of mines, with k fixed
  * k =  5, m and n random   [1000 trials]
  * k = 10, m and n random   [1000 trials]
  * k = 50, m and n random   [1000 trials]
  * k = 99, m and n random   [1000 trials]
==> passed

Test 8: check number of mines for corner cases
  * m =  5, n = 20, k =  0
  * m = 20, n =  5, k =  0
  * m =  5, n = 10, k = 50
  * m = 10, n =  5, k = 50
  * k =  0, m and n random   [1000 trials]
  * k =  1, m and n random   [1000 trials]
==> passed

Test 9: check that mines are uniformly random
  * m = 1, n = 2, k = 1 [repeated 15000 times]
  * m = 1, n = 3, k = 1 [repeated 15000 times]
  * m = 2, n = 2, k = 2 [repeated 15000 times]
  * m = 2, n = 4, k = 3 [repeated 15000 times]
  * m = 3, n = 3, k = 6 [repeated 15000 times]
==> passed

Test 10: check statistical independence of mines within an m-by-n grid
  * m = 500, n = 500, k = 125000
  * m = 500, n = 500, k = 25000
  * m = 500, n = 500, k = 225000
  * m = 100, n = 900, k = 27000
  * m = 900, n = 100, k = 63000
==> passed

Test 11: check statistical independence of mines between m-by-n grids
  * m = 1, n = 2, k = 1 [repeated 50000 times]
  * m = 1, n = 3, k = 1 [repeated 50000 times]
  * m = 2, n = 2, k = 2 [repeated 50000 times]
  * m = 2, n = 4, k = 3 [repeated 50000 times]
  * m = 3, n = 3, k = 8 [repeated 50000 times]
==> passed


Minesweeper Total: 11/11 tests passed!


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