Programming Assignment 1: Percolation — Instructions

Assignment Name	Programming Assignment 1: Percolation
Due Date	Sat 25 Aug 2012 8:59:00 PM PDT
Hard Deadline	Sun 30 Sep 2012 8:59:00 PM PDT
Submission	Go to Assignments List page to submit your solutions.
Instructions	

Specification

Here is the programming assignment specification that describes the assignment requirements.

Checklist

Here is the checklist that contains frequently asked questions and hints. If you're not sure where to start, see the section at the end of the checklist.

Web Submission

Submit a zip file named percolation.zip that contains only the two source files Percolation.java and PercolationStats.java. To zip up your source files, use one of the following three approaches:

- **Mac OS X Finder.**
  1. Select the required files in the Finder.
  2. Right-click and select *Compress 2 Items*.
  3. Rename the resulting file to percolation.zip.
- **Windows.**
  1. Select the required files in Windows Explorer.
  2. Right-click and select *Send to -> Compressed (zipped) folder*.
  3. Rename the resulting file to percolation (the .zip extension is automatic).
- **Command line (Linux or Mac OS X).**
  1. Change to the directory containing the required .java files.
  2. Execute the command `zip percolation.zip Percolation.java PercolationStats.java`

You will not receive a score or grade report unless you submit the zip file in this specified format and the source files conform to the prescribed APIs.

Assessment Report

Here is some information to help you interpret the assessment report. See the Assessment Guide for more details.

- Compilation: we compile your .java files using a Java 6 compiler. Any error or warning messages are displayed and usually signify a major defect in your code.
- Style: we run checkstyle to automatically checks the style of your Java programs. Here is a list of available Checkstyle checks, which

you can use to help decode any warning messages.

- Bugs: we run findbugs to check for common bug patterns in Java programs. A warning message strongly suggests a bug in your code but occasionally there are false positives. Here is a summary of bug descriptions, which you can use to help decode warning messages.
- API: we check that your code exactly matches the prescribed API (no extra methods and no missing methods). If it does not, no further tests are performed.
- Correctness: we perform a battery of unit tests to check that your code meets the specifications.
- Memory: we determine the amount of memory according to the 64-bit memory cost model from lecture.
- Timing: we measure the running time and count the number of elementary operations.