| COS 126 | General Computer Science | Spring 2011 |
|---|---|---|

# Programming Exam 1

This test has 1 question. You have 50 minutes. The exam is open book, open note, and open web. You may use code from your programming assignments or the Introduction to Programming in Java booksite. No communication with any non-staff members is permitted. Submit your solution via Dropbox. **Write out and sign the Honor Code pledge before turning in the test.**

*"I pledge my honor that I have not violated the Honor Code during this examination."*

**Name:** ----------

Signature

**NetID:**

| Total | |
|---|---|

| | | |
|---|---|---|
| P01 | TTh 1:30 | Keith |
| P01A | TTh 1:30 | Doug |
| P01B | TTh 1:30 | Victor |
| P01C | TTh 1:30 | Richard |
| P01D | TTh 1:30 | Gordon |
| P01E | TTh 1:30 | Arman |
| P02 | TTh 2:30 | Doug |
| P03 | TTh 3:30 | Gordon |
| P03A | TTh 3:30 | Keith |
| P04 | TTh 7:30 | Nick |
| P05 | WF 10 | Dmitry |
| P06 | WF 1:30 | Victor |
| P06A | WF 1:30 | Chris |
| P06B | WF 1:30 | Donna |
| P07 | WF 12:30 | Donna |

**Do not remove this exam from the room.**

**Problem.** Write a program that apportions $H$ U.S. House of Representatives seats to $N$ states using the *Huntington-Hill method*:

- First, assign one seat to each state.
- Assign the remaining $H - N$ seats, one at a time, by priority:
  - The *priority* of a state whose population is $p$ and currently has $n$ seats assigned is

$$\frac{p}{\sqrt{n(n+1)}}$$

  - Assign the next seat to the state with the highest priority (breaking any ties arbitrarily).

**Example.** Here is how we apportion $H = 12$ seats among the $N = 4$ states $A$ (380), $B$ (120), $C$ (310), and $D$ (190), with the populations given in parentheses. First, we assign one seat to each state. We assign the next seat to $A$ because it has the the highest priority ($380/\sqrt{2} = 268.70$). We assign the next seat to $C$ because its priority ($310/\sqrt{2} = 219.20$) is higher than that of $B$ and $D$, as well as the new priority of $A$ ($380/\sqrt{6} = 155.13$).

| *seats* | | | | *priorities* | | | | *next* |
|---|---|---|---|---|---|---|---|---|
| $A$ | $B$ | $C$ | $D$ | $A$ | $B$ | $C$ | $D$ | *seat* |
| 0 | 0 | 0 | 0 | – | – | – | – | – |
| 1 | 1 | 1 | 1 | 268.70 | 84.85 | 219.20 | 134.35 | $A$ |
| 2 | 1 | 1 | 1 | 155.13 | 84.85 | 219.20 | 134.35 | $C$ |
| 2 | 1 | 2 | 1 | 155.13 | 84.85 | 126.56 | 134.35 | $A$ |
| 3 | 1 | 2 | 1 | 109.70 | 84.85 | 126.56 | 134.35 | $D$ |
| 3 | 1 | 2 | 2 | 109.70 | 84.85 | 126.56 | 77.57 | $C$ |
| 3 | 1 | 3 | 2 | 109.70 | 84.85 | 89.49 | 77.57 | $A$ |
| 4 | 1 | 3 | 2 | 84.97 | 84.85 | 89.49 | 77.57 | $C$ |
| 4 | 1 | 4 | 2 | 84.97 | 84.85 | 69.32 | 77.57 | $A$ |
| **5** | **1** | **4** | **2** | | | | | |

Thus, $A$ receives 5 seats, $B$ receives 1 seat, $C$ receives 4 seats, and $D$ receives 2 seats.

**API specification.** Your program `HuntingtonHill.java` must be organized as a library of static methods with the following API:

| `public class HuntingtonHill` | |
|---|---|
| `double priority(int p, int n)` | *priority of a state with population p and n seats already assigned* |
| `int next(int[] populations, int[] seats)` | *index of next state to be assigned a seat given populations and current apportionment* |
| `void main(String[] args)` | *read number of seats H as a command-line argument; read N and state names and populations from stdin; write apportionments to stdout* |

**Input and output specifications.** You must read input and write output as directed below:

- *Command-line argument.* The number of seats $H$ to apportion.
- *Standard input.* An integer $N$ followed by $N$ string–integer pairs, where each pair is the name of a state and its population.
- *Standard output.* The integer $N$ followed by $N$ string–integer–integer triples, where each triple is the name of a state, its population, and its apportionment. (You need not duplicate our exact spacing.)

Assume that $H \geq N \geq 1$ and that the populations are positive integers. Here is a sample execution.

```
% more tiny.txt
4
A  380
B  120
C  310
D  190
```

```
% java HuntingtonHill 12 < tiny.txt
4
A  380  5
B  120  1
C  310  4
D  190  2
```

For convenience, the following real-world input files are available:

```
http://introcs.cs.princeton.edu/data/1790.txt
http://introcs.cs.princeton.edu/data/2000.txt
http://introcs.cs.princeton.edu/data/2010.txt
```

```
% more 2010.txt
50
Alabama           4802982
Alaska             721523
Arizona           6412700
Arkansas          2926229
California       37341989
Colorado          5044930
Connecticut       3581628
Delaware           900877
Florida          18900773
Georgia           9727566
Hawaii            1366862
Idaho             1573499
Illinois         12864380
Indiana           6501582
Iowa              3053787
Kansas            2863813
...
New_Jersey        8807501
...
Wyoming            568300
```

```
% java HuntingtonHill 435 < 2010.txt
50
Alabama           4802982     7
Alaska             721523     1
Arizona           6412700     9
Arkansas          2926229     4
California       37341989    53
Colorado          5044930     7
Connecticut       3581628     5
Delaware           900877     1
Florida          18900773    27
Georgia           9727566    14
Hawaii            1366862     2
Idaho             1573499     2
Illinois         12864380    18
Indiana           6501582     9
Iowa              3053787     4
Kansas            2863813     4
...
New_Jersey        8807501    12
...
Wyoming            568300     1
```

**Submission.** Submit the single file `HuntingtonHill.java` via Dropbox at

`https://dropbox.cs.princeton.edu/COS126_S2011/Exam1`

Be sure to click the *Check All Submitted Files* button to verify your submission.

**Grading.** *Your program will be graded on correctness and clarity (including comments). You will receive partial credit for correctly implementing the following components:*

- *The* `priority()` *function.*
- *The* `next()` *function.*
- *Reading the input data, storing it in two parallel arrays, and printing it back out.*

*You will receive a substantial penalty if your program does not compile or if you do not follow the prescribed API or input/output specifications.*