COS 126 **General Computer Science** **Spring 2013**

Programming Exam 1

This programming exam has 2 parts, weighted as indicated. The exam is semi-open: you may use the course website, the booksite, any direct links from those two, the textbook, any printed or written notes, and your past coursework on your computer. You may not use other websites. No communication is permitted, except with course staff members.

Upload your code to the dropbox. As with the COS 126 assignments, you can submit your code multiple times for testing, but only the last version will be graded.

Your code will be graded primarily on correctness. You will lose a substantial number of points if your program does not compile. However, efficiency, clarity, and code style are also factors in your grade. Remember to include headers and comments in your code. Headers MUST include your name, netID and precept number.

Print your name, netID, and precept number on this page (now), and write out and sign the Honor Code pledge before turning in this paper. It is a violation of the Honor Code to discuss this exam until everyone in the class has taken the exam. You have 90 minutes to complete the test. **Do not remove this exam from the exam room.**

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."

Pledge: _____

_____

Signature: _____

P01	12:30 TTh	Dave Pritchard
P01A	12:30 TTh	Donna Gabai
P01B	12:30 TTh	Pawel Przytycki
P02	1:30 TTh	Tom Funkhouser
P02A	1:30 TTh	Allison Chaney
P02B	1:30 TTh	Pawel Przytycki
P02C	1:30 TTh	Vivek Pai
P02D	1:30 TTh	Siddhartha Chaudhuri
P03	2:30 TTh	Tom Funkhouser
P03A	2:30 TTh	Allison Chaney
P04	3:30 TTh	Vivek Pai
P04B	3:30 TTh	Shilpa Nadimpalli
P05	7:30 TTh	Shilpa Nadimpalli
P06	10am WF	Lennart Beringer
P07	1:30 WF	Dave Pritchard
P07A	1:30 WF	Kevin Lee
P07B	1:30 WF	Siyu Liu
P08	12:30 WF	Donna Gabai
P08A	12:30 WF	Judi Israel
P09	11am WF	Judi Israel

Name: _____

NetID: _____

Precept: _____

Problem	**Score**
`FoodPart1`	/18
`FoodPart2`	/12
Total	/30

At the end of this handout, there is a listing of two datasets for the exam. The files are available online at

`http://www.cs.princeton.edu/~cos126/docs/data/Food`

and we will refer back to these examples as we describe the tasks.

The link for submitting your code is available on this term's **Assignments Page** — click on the *Submit* link for Exam 1. You can also reach it directly via the URL

`https://dropbox.cs.princeton.edu/COS126_S2013/Exam1`

Part 1: Competitive Cuisine

Every year, chefs from around the world compete to take part in the International Foodlympics. In this contest, each chef cooks a meal which is ranked on a scale from 1 to 20 by a panel of judges.

The assessments from the judges are combined into a single score by (i) discarding the maximum score, (ii) discarding the minimum score, and (iii) taking the average of the remaining scores rounded to the nearest integer. For example, if a chef is judged by 6 judges and gets scores of

$$7 \quad 14 \quad 20 \quad 12 \quad 15 \quad 16,$$

then, since 20 is the maximum and 7 is the minimum, the average computed score for this chef is $(14 + 12 + 15 + 16)/4 = 14.25$. The final score is rounded to 14.

When more than one judge enters the same minimum or maximum score, only one copy of that score is discarded. For example the raw scores

$$10 \quad 16 \quad 10 \quad 20 \quad 14 \quad 10$$

would result in an average of $(16 + 10 + 14 + 10)/4 = 12.5$ and a final score of 13 — only a single 10 (the minimum) is not counted.

In the first part of this exam, you will write a program `FoodPart1.java` with the following API:

```
public class FoodPart1
----------------------
  public static int score(int[] ratings)  // return the rounded average
                                          // ignore the min and max ratings

  public static void main(String[] args)  // read the number of countries and judges from
                                          // standard input.  print out the number of
                                          // countries.  for each country, read its name
                                          // and the judges' ratings.  then print its name
                                          // and its score.
```

Your `score()` method must not alter the array `ratings`. Note that you do not have to literally delete scores: it is enough to compute the number `J` of judges, the sum, the min, and the max; then you can use those numbers to compute the final score. You may assume $\texttt{J} \geq 3$ since otherwise the average is not defined. Note that `Math.round()` returns a `long`, but you can safely cast the value it returns to an `int` since the values will always be between 1 and 20.

The `main()` method will read data from standard input in the following format. First, there will be a line containing the number `C` of countries, a space, and the number `J` of judges. On each of the next `C` lines there will be a String — the name of that country — followed by `J` integers, with a space preceding each integer. These integers are the ratings by the judges for that country.

For example, this is a file representing three countries being judged by six judges.

```
3 6
Canada 10 16 10 20 14 10
Mexico 14 14 14 14 14 14
USA 7 14 20 12 15 16
```

You may assume that the countries' names do not contain whitespace, so you can read them with `StdIn.readString()`. You may also assume that the country names are distinct — no country sends more than one chef to the Foodlympics.

Your program should first output `C` on a line by itself, and then for each country, its name and its overall score, separated by a space. The order of countries should be the same as in the input. For the example input file above, the correct output would be

```
3
Canada 13
Mexico 14
USA 14
```

Upload your part 1 code before proceeding, to see if there are any bugs caught by the dropbox tests.
Dropbox will also run some tests directly on your `score()` method.

Continue reading on the other side of this page for the part 2 specification.

Part 2: Gourmet Glory

You will now create a second class `FoodPart2.java` whose purpose is to determine the top finishers at the International Foodlympics. The API for this part consists of two methods. The method `best()` will determine the country with the top score, set that score to zero and return the country's name. The `main()` method will use this to print out the top finishers.

```
public class FoodPart2
----------------------
  public static String best(String[] countries, int[] scores) // return the country with
                                                              // the best score, after
                                                              // setting its score to 0

  public static void main(String[] args) // read in countries and scores from standard
                                         // input, and read an integer N as a command-line
                                         // argument.  print out the N countries with the
                                         // N best scores, from best to worst.
```

Your `best()` method is required to determine the country passed in with the best score, set its score to zero, and return its name. (The `best()` method should use only the arguments passed in, and no additional static fields.) While we require that the `best()` method has the side effect of changing one value in `scores`, it should not alter any values in `countries`. You may assume that the arrays `countries` and `scores` have the same length and are in corresponding order (i.e., `countries[i]` refers to the name of the country with `scores[i]`).

If there are multiple countries *tied* for best, the first (lowest-indexed) country with that score should be deemed the winner. Return its name and set only its score to zero.

Finally, your `main()` method will read in the scores of a list of countries, and output the top `N` finishers, where the integer `N` is the first command-line argument. The input format for `FoodPart2` is identical to the output format for `FoodPart1`. (Specifically, `main()` should read, from standard input, an integer `C` representing the number of countries, and then `C` lines each consisting of a country name, followed by a space and that country's score.) Here is a sample of a valid standard input for `FoodPart2`:

```
6
GER 13
BEL 15
AUT 17
CHF 19
LUX 14
LIE 8
```

Then, where `N` is the first command-line argument, your `main()` method should print out the names of the top `N` countries. Each line should be of the form `Rank i: name` where `i` is the rank (going from 1 to `N`) and `name` is the country with that rank. For example, if the above text file were saved as `germanScores.txt`, then the output of

```
% java FoodPart2 2 < germanScores.txt
```

should be:

```
Rank 1: CHF
Rank 2: AUT
```

To do this, we suggest that your `main()` method should save the standard input data in two parallel arrays, and call the `best()` method `N` times on these arrays, printing out the returned string each time. (You may assume all input scores are greater than zero.)

You may assume that the command-line argument `N` is an integer satisfying $1 \leq$ `N` $\leq$ `C`, since the output is not well-defined otherwise.

Remember to submit your work using the dropbox link mentioned earlier.
Dropbox will also run some tests directly on your `best()` method.

Description of Dropbox Data Sets

As mentioned earlier, for your convenience, the input and output files described in this document are available at

`http://www.cs.princeton.edu/~cos126/docs/data/Food`

Sample Data Set 1

The text file `na.txt` contains the following data,

```
3 6
Canada 10 16 10 20 14 10
Mexico 14 14 14 14 14 14
USA 7 14 20 12 15 16
```

and

```
% java FoodPart1 < na.txt
```

should output

```
3
Canada 13
Mexico 14
USA 14
```

If we write this output to `naScores.txt` then

```
% java FoodPart2 3 < naScores.txt
```

should output

```
Rank 1: Mexico
Rank 2: USA
Rank 3: Canada
```

You also can compute this in one line:

```
% java FoodPart1 < na.txt | java FoodPart2 3
```

On the files page, this output is called `naRankings.txt`.

Sample Data Set 2

The text file `german.txt` contains the following data,

```
6 3
GER 1 13 20
BEL 15 15 15
AUT 17 18 3
CHF 19 7 19
LUX 14 14 15
LIE 7 8 9
```

and

```
% java FoodPart1 < german.txt
```

should output

```
6
GER 13
BEL 15
AUT 17
CHF 19
LUX 14
LIE 8
```

If we write this output to `germanScores.txt` then

```
% java FoodPart2 2 < germanScores.txt
```

should output

```
Rank 1: CHF
Rank 2: AUT
```

You also can compute this in one line:

```
% java FoodPart1 < german.txt | java FoodPart2 2
```

On the files page, this output is called `germanRankings.txt`.