Homework 2

EE 363 (Fall 2022)

Department of Electrical and Computer Engineering
Clarkson University

Instructions

Note 1: Please read all instructions carefully before submitting your work. There is 1 question in this homework for a total of **40** points.

Note 2: Solve all problems and upload your answers to Moodle using the instructions below. Make sure any code you write works on Polaris before uploading your files. You will likely lose many points if your code doesn't compile. Whenever you write solutions on paper, you need to scan all documents before submitting.

Note 3: `USR` stands for your login ID on Polaris (`polaris.clarkson.edu`). `MJR` represents your major, i.e. CE, SE, EE, CS, etc.

Note 4: In program listings and shell interactions, any **`bold, underlined, constant-width`** text indicates that it is user input.

Note 5: In program listings and shell interactions, any ***`italicized, underlined, constant-width`*** text has to be explicitly replaced by an appropriate user-supplied value.

Note 6: Do not upload any executable or intermediate files as answers to problems, unless specifically asked to do so.

Note 7: You should upload a single file named `USR_HW2_MJR.zip` to Moodle. (If you have multiple majors, please mention all of them in the file name, e.g. `USR_HW2_MJR1_MJR2.zip`.) In this ZIP file, you should have a directory named `hw2`, inside which you should have directories for each problem, named to clearly indicate the problem number, i.e. `p1`, `p2`, etc, each of which should contain the appropriate deliverable. (To verify the directory structure before creating the zip file, the `tree` command on Polaris can be used to display the directory contents in a hierarchical fashion, e.g. `tree dirName`.)

Note 8: Please read the "Note on academic honesty" in the syllabus. In the directory for *each* problem, please include a `readme.txt` file containing the following:

- It should state whether you worked alone on that probem or list the full names of all the people you discussed it with. This includes the names of the course staff, if any of them helped you with the problem.
- If your answer includes or is inpired by a non-trivial amount of content (including code) from specific sources, please cite all of them prominently.

Note 9: For this homework, you need to work individually or in groups of up to 3. There should be only one submission per group.

Note 10: On Polaris, the following command can be used to zip the contents of directory `hw2` into a file: `dest.zip`: `zip -9 -yrq dest.zip hw2`

Note that you will need to rename the ZIP file as per the instructions described earlier. Please test the generated zip file by listing its contents (`unzip -l dest.zip`) and also by extracting and viewing its contents in a separate (temporary) directory (`unzip dest.zip`).

1. [40 points] In this problem, you will write a multithreaded Java program. By implementing `Runnable`, design a class `RCompute` that can be used to compute the average of a list of numbers within a new thread.

   Study how the driver file `ClientAvgPar.java` uses `RCompute` to create multiple threads to *concurrently* compute the averages of multiple lists of numbers:

   1. The client takes three command-line arguments, viz., `MAX_LEFT` (the upper bound of the left endpoint of all the lists), `MAX_RIGHT` (the upper bound of the right endpoint of all the lists), and `NUM_LISTS` (the number of lists).
   2. every list i is specified by a left ($L_i \in \mathbb{N}$) and right endpoint ($R_i \in \mathbb{N}$) where $L_i > 0, R_i > 0$, and $L_i < R_i$, where $i \in \{1, 2, \ldots,$ `NUM_LISTS`$\}$.
   3. each `RCompute` instance is used to compute the average of the square roots of numbers in the range e.g., if $L_i = 6$ and $R_i = 20$, it will be used to compute the average of sqrt(6), sqrt(7), ..., sqrt(19), and sqrt(20).
   4. an independent thread will calculate the average of the square roots of all numbers in each range
   5. each thread prints the average it computes as shown in Listing 1.

   Given the command-line arguments `MAX_LEFT`, `MAX_RIGHT`, and `NUM_LISTS`, `ClientAvgPar` automatically generates `NUM_LISTS` pairs of end-points `L`, `R`, where `0 < L < MAX_LEFT` and `MAX_LEFT+1 < R < MAX_RIGHT` and uses them to start threads for computing averages.

   Listing 1: Sample interaction with `ClientAvgPar` where averages are computed for 8 lists of numbers. For each list, the left and right endpoints are specified in the output in parentheses. See `ClientAvgPar.java` for details.

```
$ tree --charset ascii  hw2/
hw2/
`-- p1
    |-- ClientAvgPar.java
    `-- RCompute.java

1 directory, 2 files

$ javac hw2/p1/*java

$ java hw2.p1.ClientAvgPar 75 350 8
[Thread thr_0] Average of square roots  (65 to 106) = 9.22
[Thread thr_6] Average of square roots  (12 to 263) = 11.21
[Thread thr_7] Average of square roots  (49 to 297) = 12.83
[Thread thr_4] Average of square roots  (49 to 97) = 8.50
[Thread thr_1] Average of square roots  (22 to 90) = 7.36
[Thread thr_3] Average of square roots  (13 to 136) = 8.34
[Thread thr_2] Average of square roots  (44 to 325) = 13.20
[Thread thr_5] Average of square roots  (61 to 344) = 13.91
```

You need to execute `ClientAvgPar` so that it uses your `RCompute` to average square roots in *15 lists of numbers* and store the output of this execution in `avg15.txt`. Invoke the program so that, for each list, $0 \leq L_i \leq 30$ and $30 < R_i \leq 250$.

Note: The output will in general vary with each execution because the endpoints of the lists, and hence the lists themselves, are randomly generated (in the range specified by the command line arguments).

In `readme.txt`, briefly describe how to run your code on Polaris.

Deliverable: `RCompute.java`, `avg15.txt`, `readme.txt`.

End