CIS 313, Intermediate Data Structures
Fall 2017, University of Oregon

Programming Assignment 2

Due: October 30, 2017

0 Instructions

Submit your work through Canvas. You should submit a tar file containing all source files and a README for running your project.

More precisely, submit on Canvas a tar file named lastname.tar (where lastname is your last name) that contains:

- All source files. You can choose any language that builds and runs on ix-dev.
- A file named README that contains your name and the exact commands for building and running your project on ix-dev. If the commands you provide don't work on ix-dev, then your project can't be graded and you won't receive credit for the assignment.

Here is an example of what to submit:

```
hampton.tar
  problem1-1.py
  problem1-2.py
  ...
  README
```

```
README
  Andrew Hampton

  Problem 1.1: python problem1-1.py input.txt
  Problem 1.2: python problem1-2.py input.txt
  ...
```

Note that Canvas might change the name of the file that you submit to something like lastname-N.tar. This is totally fine!

The grading for this part of the assignment will be roughly as follows:

Task	**Points**
Problem 1	25
Problem 2	25
TOTAL	50

1 Applications of the Binary Heap / Priority Queue

Problem 1. Prioritizing HTTP Requests
Suppose you are working on a web application that receives HTTP requests over a LAN in batches. When you receive a batch of requests, they have already been preprocessed with an estimate of how long it will take your application to complete each of them.

You want to serve the requests in order of the estimated service time, with the shortest requests being served first.

Additionally, your application has two tiers of service: A and B. All of the requests in the A tier should be served before any of the requests in the B tier.

Write a program that will give the correct service order according to the above criteria. Your program **must** use one or more binary heaps or priority queues to accomplish this! (You may use your language's builtin data structure. For example, in Python you should consider using either the *heapq* or *Queue* module.)

Your program should take a single command-line argument, which will be a filename. The input file will contain request strings. The first line of the input file will be an integer $0 \leq N \leq 10^6$ giving the number of requests. Following will be N lines, each containing a string having the following format:

```
    IP_ADDR TIER ESTIMATE
```

IP_ADDR is an IP address in decimal IPv4 format. TIER is either *A* or *B*. ESTIMATE is an integer $0 < X \leq 10^4$ representing a time estimate for processing the request. The separator is a single space.

Output the IP addresses in the order the requests will be served, separated by newlines. In the event of a tie in service time (and tier), the request that appears first in the input list should be served first (that is, your sort should be *stable*).

Example input file:

```
8
10.31.99.245 B 30
10.16.0.105 A 150
10.16.115.160 B 60
10.30.111.90 B 65
10.16.0.105 A 20
10.30.100.100 A 25
10.16.100.115 A 150
10.111.111.119 B 60
```

Example output:

```
10.16.0.105
10.30.100.100
10.16.0.105
10.16.100.115
10.31.99.245
10.16.115.160
10.111.111.119
10.30.111.90
```

Problem 2. Rolling Median
Suppose you have streaming (integer) data and want to compute some summary statistics. It's easy to calculate the cumulative rolling average: this is a constant time operation (look up the formula if you're interested!). What about the cumulative rolling median?

In this problem, you'll develop an algorithm to compute the cumulative rolling median and test it on simulated streaming data.

We will use the most common definition of *median*, as described on the Wikipedia page. We can simulate streaming data by giving a list of integers L of length n and calculating the median on the slice $L[0 : i]$ for every $0 < i \leq n$.

Your program should take a single command-line argument, which will be a filename. The input file will contain integers, one per line. The first line of the input file will be an integer $2 \leq N \leq 10^5$ giving the number of integers in the list L. Following will be N lines, each containing an integer $0 \leq x \leq 10^6$.

You should output the median of the slice $L[0 : i]$ for every $0 < i \leq N$, with a newline between each result. If a median is not an integer, it should be printed to one decimal place. See the sample output below.

RUNTIME: Your solution should have runtime complexity $O(n \log n)$. If you submit a slower solution, you will receive at most 40% credit for the problem.

Hint: use two binary heaps, one to hold the smaller half of the data and one to hold the larger half of the data; one of these heaps should be a minheap and one should be a maxheap.

Example input file:

```
5
1
8
4
3
2
```

Example output:

```
1
4.5
4
3.5
3
```

The first line of the sample input says that the file contains 5 integers. So, we will read in 5 integers and with each new integer compute the median of those we have seen so far.

The first integer is 1 and the median of $\{1\}$ is 1. The next integer is 8 and the median of $\{1, 8\}$ is 4.5. The next integer is 4 and the median of $\{1, 8, 4\}$ is 4. The next integer is 3 and the median of $\{1, 8, 4, 3\}$ is 3.5. The final integer is 2 and the median of $\{1, 8, 4, 3, 2\}$ is 3.