To improve the experience of customers and potentially revenue, an online music store would like to recommend music that the customers might like. Different customers have different taste. How would you customize the music recommendation for a customer?

The goal of HW4 is to recommend music for a customer based on another customer with the most similar taste. Since the store collects ratings for songs from each customer, it could find the customer with the closest ratings and recommend songs that the target customer has not rated. Each rating is between 1 (poor) and 5 (excellent). The overall algorithm has two main steps:

1. For a target customer targetCust, find the customer closestCust with the closest (discussed later) song ratings. If more than one customer are equally close, order them by their names alphabetically.

2. Recommend song(s) that closestCust gave the highest rating (≥ 4), but targetCust has not rated. For example, closestCust rated song0 and song1 with 5 and song2 with 4, and targetCust did not rate the 3 songs, song0 and song1 are recommended. For simplicity, no recommendation if closestCust did not rate any songs ≥ 4, which have not been rated by targetCust.

Let $S$ be the set of songs that both customers $x$ and $y$ have rated and $\text{rating}_x(s)$ be the rating of song $s$ from customer $x$. If $S$ is empty, $x$ and $y$ are not close at all. Otherwise, we can calculate the distance between $x$ and $y$ based on two factors: (1) the average absolute difference in ratings: \[ \frac{1}{|S|} \sum_{s \in S} |\text{rating}_x(s) - \text{rating}_y(s)|, \] and (2) the reciprocal of the number of songs they both rated: \[ \frac{1}{|S|}. \] Combining the two factors:

\[
\text{distance}(x, y) = \frac{1}{|S|} + \frac{1}{|S|} \sum_{s \in S} |\text{rating}_x(s) - \text{rating}_y(s)|. \tag{1}
\]

For simplicity, assume only 10 songs are available and they are numbered 0-9.

To manage and find the closest customer efficiently, use HeapAdaptablePriorityQueue (textbook). You may modify HeapAdaptablePriorityQueue and related classes to increase their general functionality. The program files are on the course website.

Let $n$ be $|S|$. Equation 1 takes $O(n)$ operations. If one of the ratings is changed (AddRating, EditRating or RemoveRating action), recalculating using Equation 1 would take $O(n)$ operations. This could be computationally expensive with a large number of songs and customers. In the description at the top of HW4.java:

1. discuss how *and* why you could update the distance between $x$ and $y$ using $O(1)$ operations when one rating is changed, and

2. state the method(s) in which program file(s) that implement the distance update in $O(1)$ operations and put the method(s) near the top of the program file(s).

Input: Input is from the command-line arguments for HW4.java in this order:

1. filename of initial song ratings—each line has the name of a customer and 10 song ratings. Each rating is between 1 (poor) and 5 (excellent), or 0 (not rated). The target customer is on the first line.

2. filename of actions, each line has one of the following actions:
   - AddCustomer customer
   - RemoveCustomer customer
   - AddRating customer song rating
   - EditRating customer song rating
   - RemoveRating customer song
   - GetClosestCustomer
   - RecommendSongs
   - PrintCustomerDistanceRatings

Sample input files are on the course website.

Output: Output goes to the standard output (screen), each line corresponds to an action:

- AddCustomer customer [existingCustomerError]
- RemoveCustomer customer [noCustomerError]
- AddRating customer song rating [existingRatingError]
- EditRating customer song rating [noRatingError]
- RemoveRating customer song rating [noRatingError]
- GetClosestCustomer closestCustomer/none
- RecommendSongs closestCustomer/none song1 rating1 song2 rating2 ...
- PrintCustomerDistanceRatings

PrintCustomerRatings prints the target customer first, then the customers in alphabetical order, as a table with the columns aligned (similar to the first input file). Print distance with three decimal places (xx.xxx), for example, 2.234. Print n/a as distance if it is not measured. Assume the name of a customer does not have more than 10 characters. Sample output is on the course website.

Submission: Submit HW4.java that has the main method, (modified) HeapAdaptablePriorityQueue.java and related classes, and other program files. Submissions for Individual and GroupHelp have the same guidelines as HW1.

Note the late penalty on the syllabus if you submit after the due date and time as specified at the top of the assignment.