Homework 3 Self-Evaluation

4/5 Points

2/13/2024

Attempt 1

Review Feedback
2/10/2024

Attempt 1 Score:
4/5

Add comment

Anonymous grading: **no**

Unlimited Attempts Allowed
2/15/2024

Details

Please answer the following questions about the code you submitted to the **first** submission deadline (i.e., not the resubmission, and not your work in progress for the resubmission). If you overwrote it locally, you can redownload your submission from Canvas.

For each question, answer either with the line number (or a range of line numbers) that is relevant to the question, or with "no" if your code does not do what the question is asking about. *Your answer to a question will get 1 point if your answer accurately answers the question and you did not answer "no"; your answer will get a 0 in all other cases.*

Make sure to **double-check your answer and line numbers** to make sure they are correct and also that they are referencing the correct version of the file (see above). To ensure consistency in grading across all students, **your line numbers MUST correspond to the relevant lines of code, otherwise the question will get a 0** (even if the relevant code lies elsewhere).

---

1. Do you have a unit test that checks that a key that has been `del`eleted from a hash table is not present anymore? Answer with the line where we can find this unit test.
2. Does your `AssociationList.put` method avoid storing duplicate keys? Answer with the range of lines in `AssociationList.put` where you loop over the list (or call `AssociationList.mem?`) to see whether the key is already present.
3. Hash table buckets and association lists have a lot in common! Are you reusing code between the two? Answer with a line number inside your `HashTable` class with a call to a method from your `AssociationList` class (or a call to a standalone function which both classes call). (Note: copy-pasting is not reuse! It's just bad practice.)
4. Does your `compose_phrasebook` function use structs to represent the combination of a word's translation and its pronunciation? Answer with the line where you wrote your struct definition. Using a two-element array instead would also be acceptable for this answer but not the best practice*.
5. *For this last question, answer with a short paragraph. We will not be looking for a specific correct answer, but rather for thoughtfulness and reflection.*

In the real world, many interesting data relationships do not map neatly to dictionaries' "1 key maps to 1 value" model. For example, the words in your phrasebook may be pronounced in different ways in different regions (e.g., to-MAY-to in the US vs to-MAH-to in the UK) and neither is a more or less correct dictionary value than the other. How would you adapt your phrasebook (e.g., changing keys or values, adding another ADT, etc.) to allow different pronunciations to coexist, and to be retrieved in different contexts?

*** If you're combining a fixed number of pieces of data and they all have specific, distinct meanings, then structs are better for readability and maintenance.**

View Rubric

Select Grader

Yuqi Liu (TA)

Self-Eval 22-23

Criteria	Ratings		Points
Q1 view longer description	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
Q2 view longer description	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
Q3 view longer description	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	0 / 1 pts
Q4 view longer description	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
Q5 view longer description	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
			Total points: 4

1. 302
2. 74-79
3. no
4. two-element array; 428-432
5. If the number of regions is small, the values in the dictionary can be a struct containing the English translation and the head of a linked list. Each node of the list contains two pieces of data -- the region name and the region-specific pronunciation -- and a pointer to the next node.

If the number of regions is large, the values can be structs containing the English translation and a dictionary of key (region) - value (region-specific pronunciation) pairs, such that inserting a new element in the main dictionary involves first hashing the spelling of the foreign word, then hashing the region's name.