CSCI-1200 Data Structures — Spring 2023
Lab 11 — Hash Tables

In this lab, you will experiment with our hash table implementation of a set. The key differences between the `ds_set` class (based on a binary search tree) and the `ds_hashset` class (based on a hash table, of course), are the performance of insert/find/erase: $O(log\ n)$ vs. $O(1)$, and the order that the elements are traversed using iterators: the `set` was *in order*, while the `hashset` is in no apparent order.

http://www.cs.rpi.edu/academics/courses/spring23/csci1200/labs/11_hash_tables/ds_hashset.h
http://www.cs.rpi.edu/academics/courses/spring23/csci1200/labs/11_hash_tables/test_ds_hashset.cpp

Checkpoint 1

estimate: 15-30 minutes

For the first part of this checkpoint, implement and test the `insert` function for the hashset. The insert function must first determine in which bin the new element belongs (using the hash function), and then insert the element into that bin *but only if it isn't there already.* The insert function returns a pair containing an iterator pointing at the element, and a bool indicating whether it was successfully inserted (true) or already there (false).

For the second part of this checkpoint, experiment with the hash function. In the provided code we include the implementation of a good hash function for strings. Are there any collisions for the small example? Now write some alternative hash functions. First, create a trivial hash function that is guaranteed to have many, many collisions. Then, create a hash function that is not terrible, but will unfortunately always place anagrams (words with the same letters, but rearranged) in the same bin. Test your alternate functions and be prepared to show the results to your TA.

To complete this checkpoint: Show a TA your debugged implementation of `insert` and your experimentation with alternative hash functions.

Checkpoint 2

estimate: 20-40 minutes

Next, implement and test the `begin` function, which initializes the iteration through a `hashset`. Confirm that the elements in the set are visited in the same order they appear with the `print` function (which we have implemented for debugging purposes only).

Finally, implement and test the `resize` function. This function is automatically called from the `insert` function when the set gets "too full". This function should make a new top level vector structure of the requested size and copy all the data from the old structure to the new structure. Note that the elements will likely be shuffled around from the old structure to the new structure.

To complete this checkpoint: Show a TA these additions and the test output.

Checkpoint 3

estimate: remainder of lab time

Pair up with one of your classmates for this checkpoint. Pick a problem from the test review packet on a topic that you both would like to review and strengthen your understanding. Work out the problem together on paper first. *Yes, on paper!*

Once you have completed and proofread and triple-checked your answer on paper, launch your C++ development environment and type it up. First, focus on getting it to compile. What basic, minimal testing infrastructure is necessary to get your solution to compile? What small bits of syntax were you missing or incorrect or sloppy? *Try to avoid making these mistakes on Thursday's test!*

Then, write a simple test case to exercise your solution. Compile and run the code. As time allows, fix bugs in your solution and improve and expand your test cases. If you think you've handled every possible corner case... generalize the problem (this might mean making the solution templated or removing assumptions

about the problem). Discuss the problem with a TA or mentor. You can also move on to Big 'O' Notation analysis, and stress testing your solution with very large inputs to confirm the analysis empirically. Can your improve the solution in some way? Can you re-write, re-structure the problem in a small or major way to improve the problem? (To use a different data structure or a better algorithm?)

To complete this checkpoint: When about 10 minutes remain in the lab, the TA and mentors will circulate through the room to finalize checkoffs. Show them your handwritten initial solution, and explain the typos and errors you found and debugged in working through your computer tested solution. Explain your testing, debugging, and improvements/generalizations to your initial solution and/or to the problem.