1 Goals

- Practice with hash tables;
- More practice with resizable arrays, linked lists, separate chaining.

Note that you may use whatever environment you like for this class, but your programs must be able to compile and run using `g++` (or `clang++`), `cmake`, and `make`. For this assignment, you will also need `valgrind` to check for memory errors and leaks as well as `gnuplot` for generating performance graphs. You may also find it helpful to have `gdb` for helping to debug segmentation faults. The department provides a remote development server (`ada.gonzaga.edu`) running Ubuntu that can be accessed using an ssh remote connection from within VS Code on your own computer. The remote server contains all of the tools you need for assignments in this class. Depending on your computer's configuration, you may be able to install the tools you need locally to complete the homework assignments. It is important that you start assignments early in this class so that if you have questions you can ask them and get them answered with enough time before the homework deadline (to avoid late penalties).

2 Instructions

1. Use `git clone` to clone the classroom repository created for you and to obtain the starter code (either onto the remote development server if using `ada` or locally if you are using your own machine). Be sure to frequently add, commit, and push your updated files back to your GitHub repository (via `git add`, `git commit`, and `git push`).
2. Copy your `sequence.h`, `map.h`, `arrayseq.h`, `arraymap.h`, and `binsearchmap.h` files from HW-6 into your HW-7 repository. These must be present in your repository for me to grade your submission.
3. Implement the functions declared in `hashmap.h`. See the comments and lecture notes for details.
4. Make sure your implementations pass the basic tests provided in `hw7_test.cpp`. You do not need to write any additional unit tests for this assignment (however, you are encouraged to write additional tests as you see fit). Note the tests provided are in no way comprehensive.
5. You must run `valgrind` on the `hw7_test` program to ensure it does not detect any memory issues in your implementations. You should run `valgrind` once you get the unit tests to pass. If your program has memory issues as reported by valgrind, you will only receive partial credit on the assignment.
6. Once your code is completed, all of the unit tests pass (including your new tests), and you do not have memory issues as reported by `valgrind`, run the provided performance tests in `hw7_perf`. The performance tests will generate a data file with testing results, which you will then graph using `gnuplot`. You will need to add the corresponding graphs to your assignment write up (next step). Note that running `hw7_perf` to completion should take around 10 seconds (at most) to complete. If it takes longer than this for you, there is likely something wrong with your implementation.

7. Create your assignment write up and add it as a PDF file to your assignment (GitHub) repository. You **must** save your writeup as a PDF file and name it `hw7-writeup.pdf`. (Note no capital letters, no spaces, etc.) Be sure to push all of your source code and your write up by the due date so it can be graded. (Note that you can check that everything is in your repo from the GitHub website and/or using the `git status` command.) See below for expectations concerning your assignment writeup. Once you have submitted your files to your GitHub repository and are ready to submit your assignment for grading, you must fill out the grading submission form. A link to the form will be provided in piazza (in the same post as the GitHub classroom link).

3 Additional Requirements

Details for HW-7 were discussed in class and are provided in the lecture notes. Additional suggestions and requirements are below. Note that no additional private member variables or helper functions are allowed.

Default state for HashMap. Unlike with our ArraySeq (resizable array) implementation, the default state for our HashMap is a 16-element (empty) array. Thus, your move assignment operation must leave the right-hand side object in this default state.

Table initialization. The `init_table()` helper function should set each array element to `nullptr`. You will need to initialize the table (via `init_table()` in your constructors as well as within the move assignment operator and your `resize_and_rehash()` helper function.

Copy assignment. The copy assignment operation should not rely on calls to `insert` for the copy (since this can result in unnecessary resize and rehash calls). Note that each key-value pair should be placed in the same bucket as the right-hand side object, but the order key-value pairs are placed in the buckets does not have to be preserved.

Resize and rehash. Similar to copy assignment, your resize and rehash function should not require that the new array be resized multiple times. While it is okay to call `insert` in your resize and rehash function, you must be able to guarantee that calling `insert` will not result in another call to resize and rehash.

Clearing a HashMap. The public `clear()` function should delete all key-value pairs from the hash table, however, it should not delete the hash table itself. Instead, you can think of `clear()` as simply deleting the linked list "chains".

Finding previous and next keys. Your `next_key()` and `prev_key()` functions must be worst-case $O(n)$ time. This means you cannot use sorting to implement these functions.

HashTable statistics. The min chain length function should report the minimum length of the non-empty chains. Thus, unless the table is empty, the min chain length will be 1 or larger. Similarly, the average chain length should be computed over non empty chains. These means you will need to count the number of non-empty chains and use this to compute the average chain length.

Homework writeup. Your homework write up must contain each of the graphs produced by the performance tests together with an explanation as to why you think the performance tests came out the way they did based on what you know about the implementations. Finally, briefly describe any challenges or issues you faced in completing the assignment.