Jillian James and Marika Swanberg (Our own tests)

Test name	Function name	Description	Pass/Fail
init_empty	test_init_empty	cache is initialized with zero memused	Pass
increment_memused	test_inc_memused	setting a value in cache increments memory properly	Pass
mult_inc_memused	test_mult_inc_memused	checks whether it increments memused after multiple sets	Pass
decrement_memused	test_dec_memused	setting and then deleting a value in cache decrements memory	Pass
get_valsize	test_get_valsize	get function updates valsize parameter correctly	Pass
get_general	test_get	get function returns correct value and updates valsize correctly	Pass
get_not_in_cache	test_get_not_in_cache	get function returns nullptr when asked to get entry not in the cache	Pass
get_deleted_entry	test_get_deleted	get function returns nullptr when asked to get entry that was deleted from cache	Pass
get_evicted_entry	test_get_evicted	get function returns nullptr when asked to get entry that was evicted from the cache	Pass
get_modified_entry	test_get_modified	get function returns proper value and valsize on an updated entry	Pass
evict_multiple_elements	test_set_evict	set function evicts multiple entry if necessary to make room for incoming entry	Pass
invalid_delete	test_invalid_delete	try to delete an element that is not in the cache	Pass

For Laura Yoshida and Lucas Yong:

Test name	**Pass/Fail**
init_empty	Pass
increment_memused	Pass
mult_inc_memused	Pass
decrement_memused	Fail
get_valsize	Fail
get_general	Pass
get_not_in_cache	Pass
get_deleted_entry	Pass
get_evicted_entry	Pass
get_modified_entry	Fail
evict_multiple_elements	Fail
invalid_delete	Pass

We had no compilation or linking errors with their code. It looks like they failed some of our tests that look at memused because they do not decrement memused upon deleting entries. Another problem that is probably causing most of the other fails is in the way that they store the size of the value. Their code stores that size = sizeof(val), but val is a pointer, not the actual value pointed to by the void pointer they are storing in the map. We had the same issue, so we are familiar with it.

For Sierra Schlott and Mercy Bhakta

Test name	**Pass/Fail**
init_empty	Pass
increment_memused	Pass
mult_inc_memused	Pass
decrement_memused	Pass
get_valsize	Pass
get_general	Pass
get_not_in_cache	Pass
get_deleted_entry	Pass
get_evicted_entry	Fail
get_modified_entry	Fail
evict_multiple_elements	Fail
invalid_delete	Pass

We had no compilation or linking errors. Only one couple of our tests failed, where we try to get() an entry that has been deleted from the cache. Based on the feedback from Catch2, it looks like their code throws some kind of exception in this case because the code is working with a nullptr. This situation is not defined in the API so it is not necessarily wrong that the code throws an exception. However, we looked at their source code and could not find why this exception was thrown; it looks like they are handling the case we were trying to test in their get() function, so it's hard to say what is going on. Therefore we decided to make note of this.

For Henry and Noah:
We spent about an hour reading their README.md file to try to figure out how to compile their code in a way that we could incorporate our test file. Although they had a makefule there was no documentation on how to use this in genral or how to incorporate our own test file for testing purposes. Unfortunately, we were unsuccessful and ultimately had to give up trying to sift through dozens of files in different folders and their different eviction implementations.

For Naomi Boss and Grant Quattlebaum:

Test name	**Pass/Fail**
init_empty	Pass
increment_memused	Pass
mult_inc_memused	Pass
decrement_memused	Pass
get_valsize	Pass
get_general	Pass
get_not_in_cache	Pass
get_deleted_entry	Pass
get_evicted_entry	Pass
get_modified_entry	Pass
evict_multiple_elements	Fail
invalid_delete	Pass

There were no compilation errors but the compiler gave some warnings when run with error flags: -Wall -Wextra -pedantic. The code passed all but one test. It failed the test in which we evict multiple elements from the cache in order to include a new, larger value. You need your if (memused_ > maxmem) check to be in loop, we also had this error before so we are familiar with it.