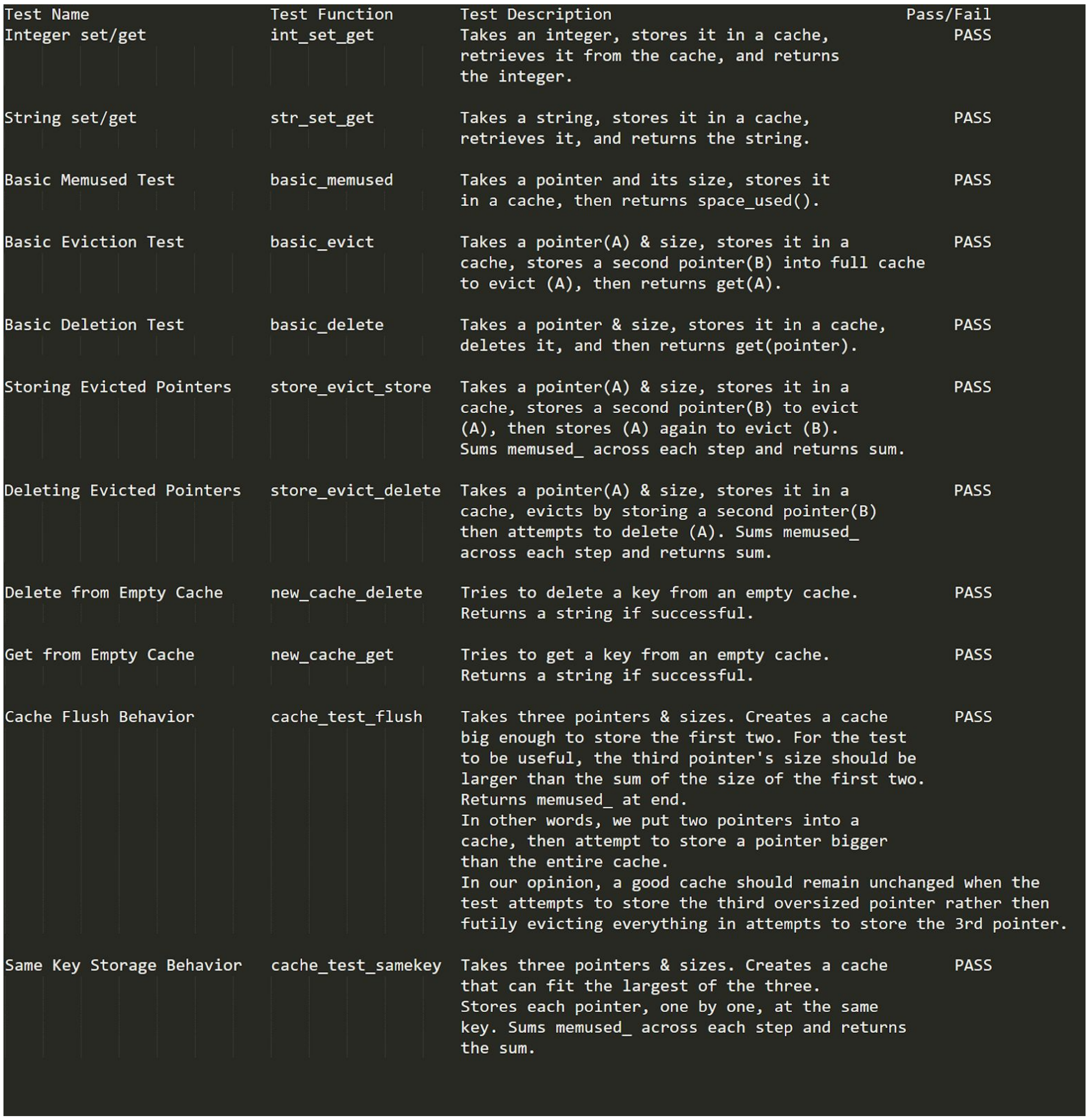
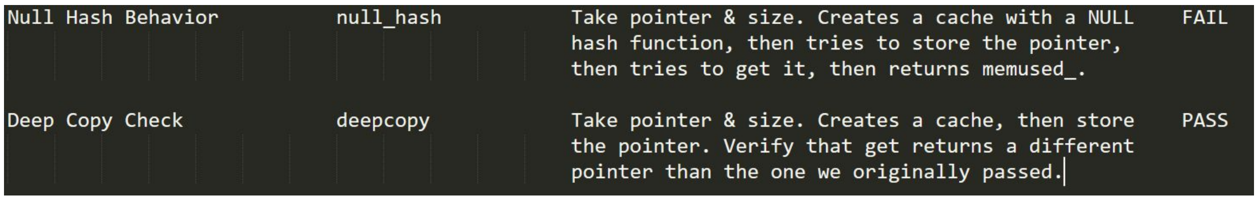
//Josh Reiss and Ryan Neumann
CS389_HW3

Part 1: Tests in cache_testy.cc



Test Name	Test Function	Test Description	Pass/Fail
Integer set/get	int_set_get	Takes an integer, stores it in a cache, retrieves it from the cache, and returns the integer.	PASS
String set/get	str_set_get	Takes a string, stores it in a cache, retrieves it, and returns the string.	PASS
Basic Memused Test	basic_memused	Takes a pointer and its size, stores it in a cache, then returns space_used().	PASS
Basic Eviction Test	basic_evict	Takes a pointer(A) & size, stores it in a cache, stores a second pointer(B) into full cache to evict (A), then returns get(A).	PASS
Basic Deletion Test	basic_delete	Takes a pointer & size, stores it in a cache, deletes it, and then returns get(pointer).	PASS
Storing Evicted Pointers	store_evict_store	Takes a pointer(A) & size, stores it in a cache, stores a second pointer(B) to evict (A), then stores (A) again to evict (B). Sums memused_ across each step and returns sum.	PASS
Deleting Evicted Pointers	store_evict_delete	Takes a pointer(A) & size, stores it in a cache, evicts by storing a second pointer(B) then attempts to delete (A). Sums memused_ across each step and returns sum.	PASS
Delete from Empty Cache	new_cache_delete	Tries to delete a key from an empty cache. Returns a string if successful.	PASS
Get from Empty Cache	new_cache_get	Tries to get a key from an empty cache. Returns a string if successful.	PASS
Cache Flush Behavior	cache_test_flush	Takes three pointers & sizes. Creates a cache big enough to store the first two. For the test to be useful, the third pointer's size should be larger than the sum of the size of the first two. Returns memused_ at end. In other words, we put two pointers into a cache, then attempt to store a pointer bigger than the entire cache. In our opinion, a good cache should remain unchanged when the test attempts to store the third oversized pointer rather then futily evicting everything in attempts to store the 3rd pointer.	PASS
Same Key Storage Behavior	cache_test_samekey	Takes three pointers & sizes. Creates a cache that can fit the largest of the three. Stores each pointer, one by one, at the same key. Sums memused_ across each step and returns the sum.	PASS



Null Hash Behavior	null_hash	Take pointer & size. Creates a cache with a NULL hash function, then tries to store the pointer, then tries to get it, then returns memused_.	FAIL
Deep Copy Check	deepcopy	Take pointer & size. Creates a cache, then store the pointer. Verify that get returns a different pointer than the one we originally passed.	PASS

Part 2: Tested code from...
(EZ & JO) Ezra Schwartz & Joe Meyer
(LU & LA) Laura Yoshida & Lucas Yong
(SI & SA) and Simon Walker-Kahne & Sam Zofkie.

(EZ & JO)
1) No compilation or linking problems were encountered.

2) TEST NAME	PASS/FAIL
Integer set/get	Pass
String set/get	Pass
Basic Memused Test	Pass
Basic Eviction Test	Pass
Basic Deletion Test	Pass
Storing Evicted Pointers	Failed
Deleting Evicted Pointers	Pass
Delete from Empty Cache	Pass
Get from Empty Cache	Pass
Cache Flush Behavior	Failed
Same Key Storage Behavior	Pass
Null Hash Behavior	Failed
Deep Copy Check	Pass

//

(LU & LA)
1) No compilation or linking problems were encountered.

2) TEST NAME	PASS/FAIL
Integer set/get	Pass
String set/get	Pass
Basic Memused Test	Pass
Basic Eviction Test	Pass
Basic Deletion Test	Pass
Storing Evicted Pointers	Failed
Deleting Evicted Pointers	Failed
Delete from Empty Cache	Pass
Get from Empty Cache	Pass
Cache Flush Behavior	Failed
Same Key Storage Behavior	Failed
Null Hash Behavior	Failed
Deep Copy Check	Failed

///

(SI & SA)
1) The source cache.cpp is using a custom cache.hh file, and needs it for compilation.
When using the cache.hh file provided to us (by EF) for HW2, this cache.cpp will not compille due to implementation of Cache.get().
More specifically, the custom cache.hh and the source cache.cpp omit every occurence of "index_type& val_size".
We compiled the source cache.cpp file with our tests in two ways:

a) We removed references to val_size from get() calls in our test file, then ran these modified tests on the normal cache.cpp

b) We added reference to val_size back into cache.cpp, then ran our normal tests on this modified cache.cpp

Both methods of compilation yield the exact same test results.

2)

TEST NAME	PASS/FAIL
Integer set/get	Failed
String set/get	Failed
Basic Memused Test	Failed
Basic Eviction Test	Pass
Basic Deletion Test	Pass
Storing Evicted Pointers	Failed
Deleting Evicted Pointers	Failed
Delete from Empty Cache	Pass
Get from Empty Cache	Pass
Cache Flush Behavior	Pass
Same Key Storage Behavior	Pass
Null Hash Behavior	Failed
Deep Copy Check	Pass

//

(SI & SA part 2)
1) Simon and Sam refined their code and asked us to test again. The source code cache.cpp no longer needs a custom cache.hh file and thus compiled without complication. Their code still fails 5 tests but it now passes String set/get and fails Same Key Storage Behavior.

2)

TEST NAME	PASS/FAIL
Integer set/get	Failed
String set/get	Pass
Basic Memused Test	Failed
Basic Eviction Test	Pass
Basic Deletion Test	Pass
Storing Evicted Pointers	Failed
Deleting Evicted Pointers	Failed
Delete from Empty Cache	Pass
Get from Empty Cache	Pass
Cache Flush Behavior	Pass
Same Key Storage Behavior	Failed
Null Hash Behavior	Failed
Deep Copy Check	Pass