1. I did not find any bugs in my final program. However, as I was implementing my HashTable,I came across some bugs because I implemented a doubly-linked list to keep track the priority of the Nodes but because of this, I forgot to connect some Nodes together every now and then.
2. All of my functions met the big-O requirements except for touch. Because I used a doubly-linked list to track precedence, in order to “touch” a key, I had to iterate through the linked list and the big-O is O(N) where N is the number of items in the HashTable.
3. bool set(const KeyType& key, const ValueType& value, bool permanent = false){
4. If key is already in the HashTable
5. If the HashTable is full

Return false

1. Otherwise,

Add the key with the given value and permanent to the HashTable

Increment the counter to how many items you have

Return true

2. Otherwise

1. If the association with the key isn’t the same

Return false

1. Update the Node so it is of highest priority
2. Return true

}

bool touch(const KeyType& key){

1. Iterate through the priority LinkedList
2. If the key of the Node is the same as key

-Link the prev and next pointers of it’s neighbors

-Update m\_head and m\_tail if needed

-Add to the front of the LinkedList a new Node with the exact same content

-Delete the current Node

-Return true

2. Return false because key was not found

}

bool discard(KeyType& key, ValueType& value){

// First deletes from the Priority linked List

1. Check if head and tail are nullptr
2. Return false
3. Move the tail pointer to it’s prev Node while using a temp Node to point to tail
4. If temp is head
5. Set head to nullptr
6. Store the values of temp’s key and value into the parameters
7. Delete temp

// Second deletes from the HashTable

1. Compute the hash of key
2. Figure out what bucket number to go to
3. Loop through the bucket’s linkedlist
4. If a Node with key and value are found

* Link it’s neighbors together
* Set the pointer in the HashTable to the next pointer if it is the same Node
* Delete the node
* Decrement numItems
* Return true

1. Otherwise, the Node was not found so return false