1. No problems that I’m aware of.

2.

HashTable() constructor satisfies the big-O requirement.

~HashTable() destructor satisfies the big-O requirement.

set() satisfies the big-O requirement.

get() satisfies the big-O requirement.

touch() satisfies the big-O requirement.

Compressor::compress() satisfies the big-O requirement.

Compressor::decompress() satisfies the big-O requirement.

3.

HashTable’s set():

Get number of bucket for given key

Check if that key is already stored in the bucket

If key isn’t stored

If bucket is full

Return false

Otherwise

Insert key/value pair into bucket

If permanent is true, modify the history to make this the most recent modification

Increase the HT’s current capacity by one

Otherwise

Update the value associated with the key to the new value

Use touch() on the node associated with the key/value pair

Return true

HashTable’s touch():

Get number of bucket for given key

If the key’s node in the bucket is not permanent

If the node is the last one modified

Do nothing

Otherwise

If the node was the first (least recently modified) item in the history

Set the first item in the history to be the one after the node

Otherwise

Link the items in the history that come before and after the node

Set the last (most recently modified) item in the history to be the current node

Return true

Otherwise

Return false

HashTable’s discard():

If there is no entry in the history

Return false

Otherwise,

Set the key and value parameters to the first node in the history’s key and value

If the node is the only node in the history

Remove all entries from the history

Otherwise

Set the first entry in the history to be the one following the current node

Delete the node from memory

Decrease the HT’s current capacity by one

Return true