Project 4 Report

1. I don’t think any of my classes have bugs.

2. Yes, my classes satisfy the big-O requirements, although my Hash Table’s discard function actually uses constant time, which is faster than the big-O requirement in the spec.

3. template <typename KeyType, typename ValueType>

bool HashTable<KeyType, ValueType>::set(const KeyType& key, const ValueType& value, bool permanent)

call another function to find the key in the hash table

if not found and hash table is not full (if full, return false)

add a new Node at end of linked list in correct bucket

if item is non-permanent

add a NodePtr object to the end of recent items list

get a ptr to this spot in the list and store in the new Node

otherwise key found

update value at Node

if Node is non-permanent

set this Node as most recent

return true

template <typename KeyType, typename ValueType>

bool HashTable<KeyType, ValueType>::touch(const KeyType& key)

if key is found and the Node has pointer to object in the recent items list

then Node must be non-permanent, so

call a function to set it as most recent

return true

otherwise return false

template <typename KeyType, typename ValueType>

bool HashTable<KeyType, ValueType>::discard(KeyType& key, ValueType& value)

if there are no items to discard

return false

get key and value of least recent item in list

(by following its pointer to the actual Node in the hash table)

delete Node in hash table by resetting pointers of previous and next items

update recent items list by resetting ptrs and deleting oldest NodePtr item