lab 18 Hash Tables with Chaining

Instructions: This lab is a practice in constructing a Hash Table with chaining. Implement a Hash Table whose constructor take an integer (the initial size of the hash table), insert, remove, and get. Hints: if the value is not found in the Hash Table return a value using the default constructor. Also, use your previous code!

```
#ifndef HASH_TABLE_H
#define HASH_TABLE_H

/* HashTable via chaining */
template<class K, class V>
class HashTable {
    private:
        /* Class to begin filling out...*/
    public:
        /* Initialize the Hash Table with size size. */
        HashTable(const int size);

        /* Deconstructor shall free up memory */
        ~HashTable();

        /* Map key -> val.
         * Return true if sucessful (it is unique.)
         * Otheriwise return false.
         */
        bool insert(const K &key, const V &val);

        /* Print out the HashTable */
        void print() const;

        /* Remove the val associated with key.
         * Return true if found and removed.
         * Otherwise return false.
         */
        bool remove(const K &key);

        /* Retrieves the V val that key maps to. */
        V& operator[](const K &key);
};

int hashcode(int key);
int hashcode(std::string &key);

#include "hashtable.cpp"
```

```
39
40 #endif
```

Write some test cases:
Create some test cases, using cxxtestgen, that you believe would cover all aspects of your code.

Memory Management:
Now that are using new, we must ensure that there is a corresponding delete to free the memory. Ensure there are no memory leaks in your code! Please run Valgrind on your tests to ensure no memory leaks!

STL:
You may use vector, queue/deque, and list from the STL. Do not use any other data structure (especially map!) Failure to follow these instructions is an automatic 0 for this lab.

How to turn in:
Turn in via GitHub. Ensure the file(s) are in your directory and then:

- $ git add <files>
- $ git commit
- $ git push

Due Date: November 11, 2019 2359

Teamwork: No teamwork, your work must be your own.