```
/* * * * * * * * * * * * * * * * * * * * * * * * * * */
// hashtablecpp.h
// A hash table implementation in C++.
// Author: Lauren E. Scott
// June 30, 2014
//
// This hash table takes in strings as data, and uses a
// division hash function to separate data into buckets.
// It makes use of my custom linked list class.
/* * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <iostream>
#include <stdlib.h>
#include <stdbool.h>
#include "linkedlistcpp.h"

using namespace std;

class HashTable {
public:
    HashTable(int s) { size = s; table = new LList<string>[size]; }
    ~HashTable() {}

    int hash(string data);
    void insert(string data);
    void print_table();
    int get_used_buckets();

private:
    int                     size;
    LList<string>*          table;
};

/* * * * * * * * * * * * * * * * * * * * * */
// Function: hash
// This function computes the index of each piece of
// data (string) inserted into the table. Specifically:
//  - The function deconstructs the string into characters.
//  - The characters' ASCII values are added together.
//  - This value is divided by the size of the table, with the remainder tak
//  - In this way, the index is never greater than the size of the table.
//  - This hash function takes O(m) time, where m is the size of the string.
/* * * * * * * * * * * * * * * * * * * * * */

int HashTable::hash(string data) {
    int string_val = 0;
    for(int i = 0; i < data.length(); i++) {
        string_val += data[i];
//      cout << "String val = " << string_val << endl;
    }
//  cout << "Final index: " << (string_val % size) << endl;
    return (string_val % size);

}

/* * * * * * * * * * * * * * * * * * * * * * */
// Function: insert
// Uses the hash function to insert data into the hash table.
/* * * * * * * * * * * * * * * * * * * * * * */

void HashTable::insert(string data) {
    int index = hash(data);
    if(&table[index] == 0) {
        LList<string> list;
        list.append(data);
        table[index] = list;
    } else {
        table[index].append(data);
    }
}

/* * * * * * * * * * * * * * * * * * * * * * */
// Function: print_table
// Prints the hash table.
/* * * * * * * * * * * * * * * * * * * * * * */

void HashTable::print_table() {
    for (int i = 0; i < size; i++) {
        cout << "Bucket " << i << ": ";
        table[i].print();
        cout << endl;
    }
}

/* * * * * * * * * * * * * * * * * * * * * * */
// Function: get_used_buckets
// Returns the number of buckets currently in use by the hash table.
/* * * * * * * * * * * * * * * * * * * * * * */

int HashTable::get_used_buckets() {
    int result = 0;
    for (int i = 0; i < size; i++) {
        if (table[i].is_empty()) {
//              cout << "Empty." << endl;
        } else {
            result++;
        }
    }
    return result;
}
```