```
#include "stdafx.h"
/* Hash.cpp
*
*  Hash table implementation from:
*  Kernighan & Ritchie, The C Programming Language,
*      Second Edition, Prentice-Hall, 1988.
*/

#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <cstring>
#include <string>

using namespace std;

#include "hash.h"



static NListPtr hashTable[HASH_TABLE_SIZE];
static int bucketSize[HASH_TABLE_SIZE];


                                    /*  Hash
                                    *  Generate hash value for string s
                                    */

unsigned Hash(string s)
{
    unsigned hashVal = 0;

    for (int i = 0; i < s.length(); i++)
        hashVal = s.at(i) +31 * hashVal;

    return  hashVal % HASH_TABLE_SIZE;
}


/*  Lookup
*  Look for s in hashTable
*/

NListPtr Lookup(string s)
{
    NListPtr np;
    int count = 0;

    for (np = hashTable[Hash(s)]; np != NULL; np = np->next)
    {
        count++;
        //if (strcmp(s, np->name) == 0)
        if (s.compare(np->name) == 0) {
            //cout << s << ": " << count << endl;
            return np;    //  found
        }
    }

    return NULL;          //  not found
}

NListPtr LookupPrintCount(string s)
{
    NListPtr np;
    int count = 0;

    for (np = hashTable[Hash(s)]; np != NULL; np = np->next)
    {
        count++;
        //if (strcmp(s, np->name) == 0)
        if (s.compare(np->name) == 0) {
            cout << s << ": " << count << endl;
            return np;    //  found
        }
    }

    return NULL;          //  not found
}

/*  Insert
*  Put (name) in hash table
*/

NListPtr Insert(string name)
{
    unsigned hashVal;
    NListPtr np;

    if ((np = Lookup(name)) == NULL)  // not found
    {
        //np = (NListPtr)malloc(sizeof(*np));
        np = new nList;
        if (np == NULL )
            return NULL;

        np->name = name;
        hashVal = Hash(name);
        np->next = hashTable[hashVal];
        hashTable[hashVal] = np;
        bucketSize[hashVal]++;
    }

    return np;
}


/*  PrintHashTable
*  Print the hash table contents
*/

void PrintHashTable()
{
    NListPtr np;

    cout << "Hash table contents:" << endl;
    cout << "--------------------\n" << endl;

    for (int i = 0; i < HASH_TABLE_SIZE; i++)
    {
        np = hashTable[i];
        while (np != NULL)
        {
            cout << setw(3) << i << ":    ";
            //cout << np->name;
            cout << endl;
            np = np->next;
        }
    }
}

void PrintHashTableBuckets()
{
    NListPtr np;

    cout << "Hash table bucket amounts:" << endl;
    cout << "--------------------------\n" << endl;

    for (int i = 0; i < HASH_TABLE_SIZE; i++)
    {
        cout << i << ": " << bucketSize[i] << endl;

    }
}
void PrintHashTableBucketsMinMax()
{
    NListPtr np;

    cout << "Hash table bucket min-max:" << endl;
    cout << "-------------------------\n" << endl;
    int max = INT_MIN;
    int min = INT_MAX;
    int empty = 0;

    for (int i = 0; i < HASH_TABLE_SIZE; i++)
    {
        //cout << i << ": " << bucketSize[i] << endl;
        if (bucketSize[i] > max) {
            max = bucketSize[i];
        }
        if (bucketSize[i] < min) {
            min = bucketSize[i];
        }
        if (bucketSize[i] = 0) {
            empty++;
        }
    }

    cout << "MAX BUCKET SIZE: " << max << endl;
    cout << "MIN BUCKET SIZE: " << min << endl;
    cout << "EMPTY BUCKETS  : " << empty << endl;
}
```