Printout for cs320-09-A18-Hash

```
// File: HashData/HashData.hpp
// Olivia Lara
// November 9, 2017

#ifndef HASHDATA_HPP_
#define HASHDATA_HPP_

#include <iostream> // ostream.
#include <string> // string.
#include <functional>
#include "ArrayT.hpp"
#include "ListL.hpp"
#include "CAMetrics.hpp"
#include "HashFunctions.hpp"
using namespace std;

// ========= HashTable =========

class HashTable {
private:
    ArrayT<ListL<CAMetricsStr> > _ht; // ht is the hash table array.

    HashFunction _hashFunction;

public:
    // Post: Creates a hash table with cap slots, initialized to be empty
    // and hash function specified by hashFunction.
    HashTable(int cap, HashFunction hashFunction) :
        _ht(cap),
        _hashFunction(hashFunction) {
    }

    // Post: Installs x in the hash table at the beginning
    // of the appropriate chain.
    void insert(CAMetricsStr const &x) {
        _ht[_hashFunction(x) % _ht.cap()].prepend(x);
    }

    // Post: Returns true if this hash table contains x.
    bool contains(CAMetricsStr const &x) const {
        return _ht[_hashFunction(x) % _ht.cap()].contains(x);
    }

    // Post: Sends the entire hash table to os. For each index of the
    // hash table that contains a nonempty chain, outputs that index
    // followed by ": " followed by the chain.
    void toStream(ostream &os) const {
        for (int i = 0; i < _ht.cap(); i++) {
          if (!_ht[i].isEmpty()) {
              os << i << ": " << _ht[i] << endl;
          }
        }
    }
};

// ========= operator<< =========

ostream &operator<<(ostream &os, HashTable const &rhs) {
    rhs.toStream(os);
```

```
// File: HashFunctions.hpp
// Olivia Lara
// November 9, 2017

#ifndef HASHFUNCTIONS_HPP
#define HASHFUNCTIONS_HPP

#include "CAMetrics.hpp"
typedef CAMetrics<string> CAMetricsStr;
typedef function<unsigned int(CAMetricsStr const &)> HashFunction;

inline unsigned int bernsteinHash(CAMetricsStr const &x) {
    string str = x.toT();
    unsigned int result = 5381;
    for (int i = 0; i < str.size(); i++) {
        result = (result << 5) + result + str[i];
    }
    return result;
}

inline unsigned int knuthHash(CAMetricsStr const &x) {
    string str = x.toT();
    unsigned int result = static_cast<unsigned int> (str.size());
    for (int i = 0; i < str.size(); i++) {
        result = ((result << 5) | ((result >> 27) & 0x000001f))^str[i];
    }
    return result;
}

#endif  /* HASHFUNCTIONS_HPP */


// new page
```