CISC 3130 Ari Mermelstein

Homework #6: Due May 17

(This absolute latest you can turn in this assignment. Any homeworks (including 1-3 and 5) turned in after that date will not be accepted.)

For this assignment, you will have to implement a map data structure using a hash table. This is good practice, since the C++ map is implemented using a balanced binary search tree (similar to what we learned in class, but more sophisticated), and in pure C++, until recently, a map using hash tables was not available.

You may assume that you always map strings to integers, and therefore you need not worry about using templates. You can also always use the same function class each time, so you don't have to templatize that either.

Here are the functions that your map must implement.

void insert (**const string&** key, **int** value);
insert this key-value pair into the map under the following rules:
If *key* already exists in the map, change its value to *value*, overwriting what was there.
If *key* is not already there, insert tis key-value pair into the map.

bool containsKey(**const string&** key);
returns true iff *key* already exists in the map.

bool containsValue(**int** value);
returns true iff value appears in the map. (N.B. Since the values need not be unique, this doesn't necessarily tell you anything interesting.)

void erase (**const string&** key);
If *key* is in the map, remove the pair that contains it as the key. (This is quite tricky).

int getValueOf(**const string&** key);
If *key* is in the map, return the associated value.
If *key* is **NOT** in the map, throw an exception.

int& operator [] (**const string&** key);

If *key* is in the map, return a reference to its associated value.

If *key* is **NOT** in the map, insert *key* with value 0, and return a reference to the newly created value.

int size();

returns the number of elements in the map

bool empty();

returns true iff the number of elements in the map is 0.

In addition, you must implement a private function called hash() that takes in a string, and transforms it into an integer value. You may also implement this as a function object like I showed in class. Either way, you must explain to me what you are doing.

You will use a hash table as the underlying structure.

The declaration for the private data member could be:

vector<list<pair<string, int> > > table;

but syntax like that becomes unreadable very quickly.

You should instead create a class called **Entry** that will store a key and a value. These member functions should be implemented.

Entry(**const string&** key, int value=0);

Creates an entry with key *key* and value *value*.

string getKey();

returns the key.

int& getValue();

returns the value.

void setValue(**int** value);

updates the value to be *value*.

Once you create this entry class, the private data member can now be:

vector<list<Entry> > table;

and this is a lot more readable code.

(If you know about inheritance in C++, you may implement an Entry to be a subclass of the STL pair.)

You must submit

MyMap.h, MyMap.cpp and useMyMap.cpp.

As always, your code must be heavily commented and documented.