

CSC 402 Fall 2010 Revision of Assignment 1, Dr. Dale Parson, due by end of October 7.

I have found problems with our g++ compiler for compiling template classes that have nested template classes such as the *kvpair* class that is nested inside class *map_chainedhashimpl*. I can get Sun's CC compiler to compile these nested classes correctly, but getting this to work in g++ is becoming too time consuming. Since templates are not the main point of this hash table exercise, I am DROPPING the following requirements from the assignment:

STUDENT PART 1 for converting the *map_* class to template classes is no longer required. All classes will continue to map (string -> integer) as they do now.

STUDENT PART 3 for moving the hash function out of class *map_chainedhashimpl* and into *map_test.cxx*. Just leave it in class *map_chainedhashimpl*.

STUDENT PARTS 4 and 5 are no longer required because they are part of STUDENT 3 having to do with moving the hash function out of class *map_chainedhashimpl*.

HERE ARE THE PARTS THAT YOU MUST DO:

STUDENT 2: Change "typedef vector<kvpair> bucket" to "typedef list<kvpair> bucket" and change any use of vector in this file or in *map_chainedhashimpl.cxx* THAT DEPENDS ON THIS typedef with list. You can insert new elements at either the front or the back of the doubly-linked STL list, but in order to match prior test output from vector you should insert at the back. Read all appropriate STL docs in both <http://cplusplus.com/reference/stl/> before commencing work.

STUDENT 6: Your hash function needs to improve on the behavior of the prior hash function. Looking at *testchainedhash3.ref* reveals the problem. While the current hash function does use all the bits of information in the string keys, it overlays them at the bottom end of the hash bucket index bits. Looking at *testchainedhash3.ref* reveals clustering of collisions in buckets 16 through 47 (32 buckets). The remaining 480 buckets at the end of the test run are empty, and the MAX bucket size is 51. Ideally for 512 buckets in a hash table of 1012 unique keys, density should not exceed 2 entries per bucket.

After making your changes to the hash function, you will have to re-examine the *test1* and *test2* .out files to make sure that they are correct, and then copy them to the corresponding .ref files. For the *test3.out* file make sure that you have improved the MAX bucket size by improving the hash function. You should still have 1012 unique keys, but the MAX bucket size must be < 51, ideally much less. You do not need to ensure that every bucket in *test.out* is correct, as long as you do not lose any entries and the MAX bucket size decreases. You must check for complete correctness of *test1.out* and *test2.out* before replacing the .ref files.

Note the file names below. Copy a .out to its .ref file ONLY when you are sure that the new .out file is correct as outlined above! You will lose points if the .ref file is buggy. Changing the hash function will change the .out file. When it is correct, copy it to the .ref file.

OTHER NOTES:

Please add a line between these two lines in the makefile:

```
diff $(TARGET).out $(TARGET).ref > $(TARGET).dif
diff $(TARGET)3.out $(TARGET)3.ref > $(TARGET)3.dif
```

so that you have this:

```
diff $(TARGET).out $(TARGET).ref > $(TARGET).dif
diff $(TARGET)2.out $(TARGET)2.ref > $(TARGET)2.dif
diff $(TARGET)3.out $(TARGET)3.ref > $(TARGET)3.dif
```

The first time you run test 2 you will see this difference in testchainedhash2.dif

```
1c1
< TEST2
---
> hash
```

If this is your only difference in those files, simply do this to update the reference file:

```
cp testchainedhash2.out testchainedhash2.ref
```

You will have to do similar steps FOR ALL THREE REFERENCE FILES after you complete STUDENT STEP 6 above. Changing the hash function will change the layout of these files. You must manually inspect your .out files to make sure that they have the correct keys and values before copying each .out file to its corresponding .ref file.

Files testchainedhash3.out and testchainedhash3.ref are big, so you can just make sure that the number of key -> value mappings does not change, in addition to the requirements of STEP 6.

Finally, if you are getting a lot of compilation errors scrolling off of the screen at any point, do this:

```
gmake clean build > junk 2>&1
```

and then edit file "junk" to see the compiler errors. The above command line redirects the output of the compilation steps from the makefile into a temporary file called junk that you can inspect.

HERE IS AN ADDITIONAL REQUIREMENT FOR THIS ASSIGNMENT

STUDENT 7: Add the following lines of code to the bottom of method `map_chainedhashimpl::dump()` after the reporting of the MAX (included below):

```

cout << "MAX bucket size is " << maxbucketsize << endl ;
double mean = (double) count / (double) capacity ;
cout << "MEAN bucket size is " << mean << endl ;
double sumsquares = 0.0 ;
for (int i = 0 ; i < capacity ; i += 1) {
    double lengthDiff = (table[i] == NULL) ? 0 : table[i]->size();
    lengthDiff = lengthDiff - mean ;
    sumsquares = sumsquares + (lengthDiff * lengthDiff);
}
cout << "STDDEV = " << sqrt(sumsquares / (double) capacity) << endl ;

```

You must “#include <math.h>” near the top of the file for the sqrt() function.

In addition to reporting the maximum bucket size, this dump() code reports the mean (average) and standard deviation for bucket sizes. The mean is not a useful measure of the hash function, because the mean bucket size is simply the number of keys divided by the number of buckets. However, the standard deviation shows the variability among bucket sizes. The lower this number, the better the hash function, because a low standard deviation indicates that the key -> value mappings are not concentrated in a relatively few, large buckets.

Here are the measures of bucket size for test 3 before my improvements to the hash function.

```

-bash-3.00$ grep MAX testchainedhash3.ref
MAX bucket size is 51
-bash-3.00$ grep MEAN testchainedhash3.ref
MEAN bucket size is 1.97656
-bash-3.00$ grep STDDEV testchainedhash3.ref
STDDEV = 7.90986
-bash-3.00$ grep EMPTY testchainedhash3.ref | wc -l
480

```

Here are the measures of bucket size for test 3 after my improvements to the hash function.

```

-bash-3.00$ grep MAX testchainedhash3.ref
MAX bucket size is 9
-bash-3.00$ grep MEAN testchainedhash3.ref
MEAN bucket size is 1.97656
-bash-3.00$ grep STDDEV testchainedhash3.ref
STDDEV = 1.43186
-bash-3.00$ grep EMPTY testchainedhash3.ref | wc -l
64

```

Your improved hash function must show a reduction in MAX bucket size, a reduction of STDDEV and a reduction in the number of EMPTY buckets. You need to get at least half way to my numbers to earn better than 92%! Your hash function does not need to show improvement for tests 1 and 2 (testchainedhash.out and testchainedhash2.out) because the amounts of data in these tests are not statistically significant. Run **gmake turnitin** from your project directory.