ECS 60 **Programming Assignment #3** **Winter 2017**

Due: Wednesday, February 15th, Write-up by 4:00 P.M., Program at 11:59 pm to p3 in the cs60 account.
Filenames: authors.csv, gen.cpp
Format of authors.csv: author1_email,author1_last_name,author1_first_name
author2_email,author2_last_name,author2_first_name

#1 (40 points) I have written an extended version of the timetest.cpp from programming assignment #1, called timetest3.cpp. Both the source code, and PC executable are available in ~ssdavis/60/p3. For this question, you are to write an extensive paper (4-7 pages typed double spaced and no more, not including the tables) that compares the performance of eight new ADTs and SkipList for the four files, File1.dat, File2.dat, File3.dat, and File4.dat. You need to run each new ADT only once on each file. If an ADT does not finish within five minutes, then note that and kill the program. For BTree try M = 3 with L = 1; M = 3 with L = 200; M = 1000 with L = 2; and M = 1000 with L = 200. For the Quadratic Probing Hash try load factors of 2, 1, 0.5, 0.25, and 0.1. For the Separate Chaining Hash try load factors of 0.5, 1, 10, 100, and 1000. To set the load factor for hash tables in timetest3, you supply the size of the original table. For example, for File1.dat to have a load factor of 5 you would enter 500000 / 5 = 100000 for the table size.

Each person must write and TYPE their own paper. There is to be NO group work on this paper. There are two ways to organize your paper. One way is by dealing with the results from each file separately: 1) File1.dat, 2) File2.dat, 3) File3.dat, 4) File4.dat, and 5) File2.dat vs. File3.dat. If there are differences between how a specific ADT performs on File2.dat and File3.dat explain the differences in the last section. The other way is to deal with each ADT separately.

In any case, make sure you compare the trees (including the BTree using M = 3 with L = 1) to each other and to skip list. For BTrees, explain the performance in terms of M and L. You should also compare the hash tables to each other somewhere in your paper. For the hashing ADTs, you should also discuss the affects of different load factors on their performance with each file. Compare the performance of the Quadratic Probing hash with QuadraticProbingPtr in a separate paragraph. You should determine the big-O's for each ADT for each file; this should include five big-O values: 1) individual insertion; 2) individual deletion; 3) entire series of insertions; 4) entire series of deletions; and 5) entire file. Use table(s) to provide the run times and the big-O's.

Do not waste space saying what happened. The tables show that. Spend your time explaining what caused the times that were they were relative to each other. Always try to explain any anomalies you come upon. For example, for most ADTs, you should clearly explain why there are different times for the three deleting files. While a quick sentence explaining the source of a big-O is enough for ADT-File combinations that perform as expected, you should devote more space to the unexpected values.

Five points of your grade will be based on grammar and style. If you use Microsoft Word, go to the menu Tools:Options:Spelling &Grammar:Writing Style and set it to Formal. This setting will catch almost all errors, including the use of passive voice.

#2 (1.5 hours, 10 points) Filename: gen.cpp.

In this project, you will create a cross reference generator. You can assume that you have a file of text. The name of the file will be passed as the sole parameter to your executable. You want to read in the file and print out an alphabetized list of all the words in the file, using the following format: word, number of occurrences, lines that the word appeared on. To keep things simple for this assignment, a "word" is defined as a series of consecutive alphabetic characters. You should use the isalpha() function of the <cctype> header to determine what is, and is not an alphabetic character. You may assume that your cross-reference generator is not case-sensitive, that is the words "Did" and "did" are the same, and printed out in the solution as "did". Your program should accommodate an unlimited number of occurrences of words. Your program file should be named gen.cpp. While your program will not be judged on time, testhw will only allow 2 seconds for it to process shakespeare.txt.

The output must match that of my gen.out. My format uses setw(18) for the word, then the number of occurrences, then three spaces, and then a list of the line numbers.

The only additional files you may use are the Weiss files available in p3. You should submit ALL Weiss files that your program is dependent upon. You may modify the Weiss files, but not #include any new headers in them. Don't forget dsexceptions.h!!! You may not use the STL, nor #include any system header file than cctype, cstdlib, cstring, iostream, iomanip, and fstream. Note that testhw will just "g++ gen.cpp –o gen.out" to create your executable.

For example, running the cross-reference generator with the last two lines from the Gettysburg Address

That the nation, shall have a new birth of freedom, and that government of the
people by the people for the people, shall not perish from the earth.

```
[ssdavis@lect1 p3]$ gen.out gettyshort.txt
a                 1   1
and               1   1
birth             1   1
by                1   2
earth             1   2
for               1   2
freedom           1   1
from              1   2
government        1   1
have              1   1
nation            1   1
new               1   1
not               1   2
of                2   1,1
people            3   2,2,2
perish            1   2
shall             2   1,2
that              2   1,1
the               5   1,1,2,2,2
[ssdavis@lect1 p3]$

[ssdavis@lect1 p3]$ gen.out shakespeare.txt > temp
[ssdavis@lect1 p3]$ head -20 temp | tail -10
abatements        1   27527
abates            1   104389
abbess            25
13268,14935,14937,14943,14948,14956,14958,14960,14962,14969,14993,14995,15003,150
13,15019,15039,15062,15072,15193,15195,15245,15247,15257,15267,15313
abbey             20
15024,15031,15061,15071,15098,15176,15178,15191,15314,35214,39123,43893,52363,524
70,56156,56284,56342,97262,120046,120061
abbeys            1   53570
abbominable       1   65398
abbot             9   52470,52472,87985,90144,90487,90489,90494,90944,91153
abbots            1   54894
abbreviated       1   65397
abed              8   5915,11006,21435,67281,85708,98048,98706,115766
[ssdavis@lect1 p3]$
```