

Lab 4 COSC 3319 Spring 2018 Burris

Due: This lab will not be accepted after Thursday April 12, 2018.

For your convenience a file of 200+ random words and names (Words200D16) has been provided on Blackboard. Each line of text in the file is exactly 16 characters in length with the word left justified. You may utilize any code developed in class directly in the lab. While not initially apparent, this lab will be more of a thought problem than coding problem. A careful examination of the sample code utilized in class will reveal almost all code required for the lab has already been developed. Indeed, with so many examples, *I assume you should be able to complete the "A" option code with little trouble. Hence grading will emphasize your evaluation and presentation of results the correctly coded results. Be professional grade!*

"C" Option:

Assume a main memory hash table of size 128. The hash address is to be calculated using folding combined with the division remainder technique. The hash address must be computed by ((adding characters 3-4 of the key to characters 8-9 of the key) * 256 + the first character of the key) modulo 128. All characters must be treated as positive integers in all steps of the calculation! Use the remainder as the hash address. For example, if the string contains the value "ABCDEFGHIJK45678" then the hash address would be $HA \leftarrow ((\text{"CD"} + \text{"HI"}) * 256 + \text{"A"}) \text{ modulo } 128$. I have never used this particular function but anticipate it will be substantially less than optimal generating a plethora of collisions. Remember all characters and hash addresses must be treated as positive numbers. I recommend using a long_integer to hold the intermediate results of the hash calculation. Remember a "slice" in Ada allows the user to treat a substring as a single unit. Given str has the value "ABCD" str(2..3) refers to the characters in positions 2 through 3) as a group from the string str, i.e., "BC." I am assuming you have instantiated a function which converts a two character (8 bits per ASCII character) to a long integer (32 bits) so overflow will not occur in the sum.

Each entry placed in the hash table must be a record with 3 fields. The first field is the key, the second the initial hash address, and the third field is the number of probes to insert the key in the table counting the initial hash address as the first probe.

- A) Create a hash table in main memory and fill it 50% full. Use the linear probe technique developed in class to handle collisions. Now look up the first 30 entries placed in the table. Print the minimum, maximum, and average number of probes required to locate the first 30 keys placed in the table. Now search for the last 30 keys placed in the table. Print the minimum, maximum, and average number of probes required to locate the last 30 keys placed in the table. Print the contents of the hash table clearly indicating open entries (this should allow you to see the primary/secondary clustering effect). Calculate and print the theoretical expected number of probes to locate a random item in the table. Explain your empirical

results in light of the theoretical results. **Your grade points will be highly dependent on your explanation!**

- B) Create a hash table in main memory and fill it 90% full. Use the linear probe technique developed in class to handle collisions. Now look up the first 30 entries placed in the table. Print the minimum, maximum, and average number of probes required to locate the first 30 keys placed in the table. Now search for the last 30 keys placed in the table. **YOU MUST PHYSICALLY SEARCH FOR EACH KEY TO CALCULATE THE STATISTICS!** Print the minimum, maximum, and average number of probes required to locate the last 30 keys placed in the table. You must start filling the table with the same keys in the same order as used when only filling the table 40% full. Print the contents of the table clearly indicating open entries (this should allow you to see the primary clustering effect). Calculate and print the theoretical expected number of probes to locate a random item in the table. Discuss your empirical results in light of the theoretical results. **Your grade points will be highly dependent on your explanation!**
- C) Repeat A and B above but use the random probe for handling collisions as developed in class. When you print the contents of the table you should be able to visually see the secondary clustering affect. Calculate and print the theoretical expected number of probes to locate a random item in the table. Discuss your empirical results in light of the theoretical results. **Your grade points will be highly dependent on your explanation!**
- D) **Present all your results in a neat tabular format (typed naturally). Compare your results to the theoretical results. If there are differences, please explain why. I expect you to physically calculate the theoretical values for comparison to your empirical results for both the linear and random probes!**
You may not ask a friend, you may not look them up in some table! Providing someone else with the results will not only cause them to fail the lab but you as well. You may however teach them to use a calculator, spreadsheet, log tables, or other techniques.
- E) My required hash function has multiple weaknesses. Criticize the hash function on a technical basis. **To receive full credit, you must state clearly and explicitly state why my hash function should fail!** Based on your criticism, write a better hash function. **You must explain explicitly why your hash function should be better from both a theoretical and empirical standpoint. You will not receive credit for simply writing a hash function that performs better.** Implement the hash function and generate the same results as required for parts A thru D presented in tabular format for comparison. Formally evaluate the results as part of your lab (typed evaluation). **The results for all parts of the lab should appear in a single table.** Shame on you if your hash function's performance does not exceed the theoretical values for both the linear and random probe collision handling techniques. Script Kiddies may have trouble evaluating the failures of my hash function and creating a better function.

"B" Option:

You need not complete the "C" option using a main memory table. Rather, complete the "C" option **using a random access (relative) file for the hash table**. The hash table must never appear in main memory! Be sure to address all parts the "C" option.

A relative file is preferred for this option. You may however use a commercial database product accessed from your "C," Java, COBOL, or Ada program if desired rather than a relative file. If a database is utilized, the primary key must be the hash address calculated as specified above.

"A" Option:

Do both the "C" and "B" options. **Professionally compare the results.**