Lab 5 COSC 3319 Fall 2012 Burris

Due: Friday December 7, 2012 (Last Class Day) *This lab will not be accepted late!*

For your convenience a file of 200 random English words has been provided at on Blackboard. Each line of text in the file is 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 most of the grade will be based on your evaluation and presentation of results.*

**"C" Option:**

Assume a main memory hash table of size 256. The hash address is to be calculated using folding combined with the division remainder technique. Folding **must** be accomplished by adding characters 15 and 16 treated as a 16 bit integer to characters 12 and 13 treated as a 16 bit integer. For example, assume the word is “123456789012abcd.” After folding, the sum will be “cd” + “2a.” Compute the final hash address as the remainder of the sum divided by 256. In English (not executable code), the folding step for a 16 character string “str” would be sum = abs(StringToInteger(str(15..16)) + abs(StringToInteger(str(12..13)). The notation str(15..16) is called a “slice” in Ada and refers to the characters in positions 15 through 16 of the string “str.” I am assuming you have instantiated a function which converts a two character ASCII string to a long integer (32 bits) so overflow will not occur in the sum. The hash address is the sum modulo the table size 256.

A) Create a hash table in main memory and fill it 60% 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 items 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 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 largely determined by the 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 items placed in the table. 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 60% 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 largely determined by the 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 largely determined by the 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 you do not like 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, spread sheet, log tables, or other technique using different values.

E) My required hash function has multiple weaknesses. Criticize the hash function on a technical basis. **To receive full credit, you must state explicitly why my hash function failed!** Based on your criticism, write a better hash function. **Explain explicitly why your hash function is 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 C 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 you write a hash function whose performance exceeds the values calculated for random data!

**"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 questions raised in the “C” option.

You may 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:**

Choice (do one of the following only):

A) Do both the "C" and "B" options.

B) First complete all requirements for the "B" option. Secondly enter 150 keys into a hash table of size 100 using either main memory or a random access file. Place all keys that collide on a linked list treated as a circular queue, i.e., use the 100 entries in the hash table as list heads to linked lists of all keys that hash to that address.

Now look up the first 60 entries placed in the table. Print the minimum, maximum, and average number of probes required to locate the first 60 keys placed in the table. Now search for the last 60 items placed in the table. Print the minimum, maximum, and average number of probes required to locate the last 60 keys placed in the table. Exhibit your results in a tabular format. Evaluate this strategy for handling collisions. What are its strengths and weaknesses? Compare your results to the linear and random probe techniques.