Laboratory 10: Cover Sheet

Name ___Henry Huffman_________________ Date __10/28/14_______

Section _________1001________________________

Place a check mark in the *Assigned* column next to the exercises your instructor has assigned to you. Attach this cover sheet to the front of the packet of materials you submit following the laboratory.

Activities	**Assigned:** Check or list exercise numbers	**Completed**
Implementation Testing	✓	
Programming Exercise 1		
Programming Exercise 2		
Programming Exercise 3		
Analysis Exercise 1	x	
Analysis Exercise 2	x	
	Total	

Name ______________________________________ Date _____________________

Section _____________________________________

Check with your instructor whether you are to complete this exercise prior to your lab period or during lab.

Test Plan 10-1 (Hash Table ADT operations)

Test case	**Commands**	**Expected result**	**Checked**

Laboratory 10: Programming Exercise 1

Name ____ Date ______________________

Section _______________________________________

Test Plan 10-2 (Login Authentication Program)		
Test case	**Expected result**	**Checked**

Laboratory 10: Programming Exercise 2

Name ______________________________________ Date _____________________

Section _____________________________________

Test Plan 10-3 (perfect minimal hash operation)		
Hash formula	**Expected result**	**Checked**

Laboratory 10: Programming Exercise 3

Name ______________________________ Date ____________________

Section ______________________________

Discuss the results with your lab instructor.

Test Results Table 10-4 (stdDeviation operation)

Hash function used	**Expected distribution quality (good/fair/poor)**	**Standard deviation**	**Measured relative distribution quality**
Hash Algorithm #1 `return 0;`			
Hash Algorithm #2 `return int(str[0]) * 10 + str.length();`			

Hash Algorithm #3
`double val = 0;`
`for (int i=0;`
`     i<str.length();`
`     i++)`
`{ val += (val*1.1)*str[i];}`
`return int(val);`			

Hash Algorithm #4			
Hash Algorithm #5			

Laboratory 10: Analysis Exercise 1

Name ______Henry Huffman_________________ Date ______10/28/14____

Section ___________1001_______________________

Given a hash table of size *T*, containing *N* data items, develop worst-case, order-of-magnitude estimates of the execution time of the following Hash Table ADT operations, assuming they are implemented using singly-linked lists for the chained data items and a reasonably uniform distribution of data item keys. Briefly explain your reasoning behind each estimate.

insert O(n)

Explanation:

Assuming that the implementations are using singly-linked lists, the insert function must traverse every single node to correctly input the value.

retrieve O(n)

Explanation:

Again, assuming that the implementations are suing singly-linked lists, the retrieve function may have to traverse each node before finding the correct value, if the value is found at all.

What if the chaining is implemented using a binary search tree instead of a singly-linked list? Using the same assumptions as before, develop worst-case, order-of-magnitude estimates of the execution time of the following Hash Table ADT operations. Briefly explain your reasoning behind each estimate.

insert O($\log_2(n)$)

Explanation:

With binary search trees, even if the node we are searching for is the very last leaf of the tree, each search is reduced by checking each branch. However, if there are only nodes to the left or two the right, the binary search tree will form a singly-linked list anyways, which may result in the magnitude of n.

retrieve O($\log_2(n)$)

Explanation:

This function is similar to the last. There is a potential to remove unneeded searches by comparing branches. However, if the tree arranges its data in a singly-linked list representation by only having nodes to the left and/or right of the root, the resulting search magnitude may be n.

Laboratory 10: Analysis Exercise 2

Name ____Henry Huffman___________________ Date ___10/28/14___________

Section ___1001________________________________

Part A

For some large number of data items—e.g., *N*=1,000,000—would you rather use a binary search tree or a hash table for performing data retrieval? Explain your reasoning.

A hash table because it will greatly reduce the number of searches needed to perform a retrieval. For instance, assuming the worst-case scenario of 1,000,000, the binary search tree would have to traverse a more branches in order to locate the specified data node. With a hash table, the number of searches are drastically reduced because each tree will only contain the values that categorized them into the hash. Therefore, there will be less searches using the hash table.

Part B

Assuming the same number of data items given in Part A, would the binary search tree or the hash table be most memory efficient? Explain your assumptions and your reasoning.

During runtime, the hash table would be much more efficient because the number of activation frames needed to conduct each operation is less than the activation frames needed for such a large binary search tree. In terms of memory allocation without the accounting for the activation frames, the binary search tree is less.

Part C

If you needed to select either the binary search tree or the hash table as the general purpose best data structure, which would you choose? Under what circumstances would you choose the other data structure as preferable? Explain your reasoning.

For general use, the use of a binary search tree would suffice. The hash table is preferable when the number of data items is significantly large. This is true because it helps reduce the number of searches or traverses needed to manipulate the data.