Laboratory 10: Cover Sheet

Name _Leah Kramer_____ Date 11/08/2017____

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	✓	✓
Analysis Exercise 2	✓	✓
	Total	

Laboratory 10: Implementation Testing

Test Plan 10-1 (Hash Table ADT operations)			
Test case	**Commands**	**Expected result**	**Checked**
Insert	+f +m +leah +leroy +z +l +x +r +s +a +u +v +e +olivia +h +k +b +n +q +p +i +c +d +g	0: b i olivia p 1: c q x 2: d k leroy r 3: e l s z 4: f leah m 5: g n u 6: a h v	✓
Remove	-i -leah -q -s -k -g -v	0: b olivia p 1: c x 2: d leroy r 3: e l z 4: f m 5: n u 6: a h	✓
Remove (invalid)	-q	Could not remove data item with key (q)	✓
Retrieve	?olivia	Retrieved data item with key (olivia) and value (134)	✓
Retrieve (invalid)	?s	Could not retrieve data item with key (s)	✓
Retrieve (invalid)	?leah	Could not retrieve data item with key (leah)	✓
Empty (not empty)	e	Hash table is NOT empty	✓
clear	c	Clear the hash table 0: 1: 2: 3: 4: 5: 6:	✓
Empty (is empty)	e	Hash table is empty	✓
Remove (empty)	-leroy	Could not remove data item with key (leroy)	✓
Retrieve (empty)	?leroy	Could not retrieve data item with key (leroy)	✓

Laboratory 10: Programming Exercise 1

Test Plan 10-2 (Login Authentication Program)		
Test case	**Expected result**	**Checked**
jack broken.crown	Authentication Successful	✓
jill tumblin’down	Authentication Successful	✓
mary contrary	Authentication Successful	✓
bopeep sheep!lost	Authentication Successful	✓
simon no!pieman	Authentication Successful	✓
cole merry-soul	Authentication Successful	✓
jack jack	Authentication failure	✓
jack broken.crrrrown	Authentication failure	✓
	Authentication failure	✓
	Authentication failure	✓

Laboratory 10: Analysis Exercise 1

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): Worst case for linked list is O(1), but for hash is O(n). Take the greatest order of magnitude for the worst case.

retrieve O(n): Explanation: Worst case for linked list is O(n), but for hash is O(n). Take the greatest order of magnitude for the worst case.

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(n): Explanation: Worst case for BST and has is O(n). Take the greatest order of magnitude for the worst case.

retrieve O(n): Explanation: Worst case for BST and has is O(n). Take the greatest order of magnitude for the worst case.

Laboratory 10: Analysis Exercise 2

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.
I would prefer to use a hash table. Worst case scenarios for BST and hash are the same, O(n), but hash tables have better average case scenarios, which are O(1), than BST, which are O(log(n)).

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.
A binary search tree is more efficient than hash tables when it comes to memory usage. Binary search trees only allocate the memory they need as they need it, where a hash table allocates an entire array for what it assumes it will need.

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.

I would use a hash table as a general purpose data structure. An instance in which I would use a BST instead would be if ordered traversals are needed. Due to the way hash tables store data, it would be highly inefficient to access elements in order.