Laboratory 10: Cover Sheet

Name: Ernest Landrito Date: November 5, 2013

Section: 1

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

Name: Ernest Landrito Date: November 5, 2013

Section: 1

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**
Insert Retrieve Remove	+1 +2 +3 ?1 ?2 ?3 -1 -2 -3 q	Inserted 1 2 3 Retrieved 1 2 3 Removed 1 2 3 quit	

Laboratory 10: Programming Exercise 1

Name: Ernest Landrito Date: November 5, 2013

Section: 1

Test Plan 10-2 (Login Authentication Program)		
Test case	**Expected result**	**Checked**
jack jill jack broken.crown jack broken.crrrrown	0: 1: jack mary 2: 3: bopeep cole jill 4: 5: 6: simon 7: Login: Password: Authentication failure Login: Password: Authentication successful Login: Password: Authentication failure Login:	

Laboratory 10: Analysis Exercise 1

Name: Ernest Landrito Date: November 5, 2013

Section: 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)

Explanation: Because accessing a location of an array is in constant time, the insert function is directly dependent on the insert function of the List thus making the order of magnitude N having the data be inserted at the bottom of the list at the array location, given all the values hashed to the same location.

retrieve O(N)

Explanation: Likewise, accessing the location of the array is constant and then the order of magnitude is based on the retrieve function of the list. Worst case scenario, all the values hashed to one array location and the value is at the end of the list.

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(log2(n))

Explanation: Because accessing a location of an array is in constant time, the insert function is directly dependent on the insert function of the BST thus making the order of magnitude log2(n) having the data be inserted at the bottom of the list at the array location, given all the values hashed to the same location.

retrieve O(log2(n))

Explanation: Likewise, accessing the location of the array is constant and then the order of magnitude is based on the retrieve function of the list. Worst case scenario, all the values hashed to one array location and the value is at the bottom of the tree which in turn takes log2(n) time to get to.

Laboratory 10: Analysis Exercise 2

Name: Ernest Landrito Date: November 5, 2013

Section: 1

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 rather use a binary search tree because a hash table is best used when you could have as little collision as possible and having such a large number of data items would either cause a high chance of collision or a extremely large array which are both not ideal.

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.

The binary search tree would be more memory efficient because in having a hash table, you would need a large array to cause little collision.

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.

General purpose best data structure would be a hash table because if the hash table has minimal collisions, then the search insert and delete functions would be in constant time. I would choose a Binary search tree in situations where I need to find max and minimum keys often. This would be the case because in a BST it is one function to find the minimum key where as the hash table would have to hash through every location.