Laboratory 13: Cover Sheet

Name Duncan Wilson Date February 18th, 2014

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	X	
Programming Exercise 1	X	
Programming Exercise 2	X	
Programming Exercise 3	X	
Analysis Exercise 1		
Analysis Exercise 2		
	Total	

Name Duncan Wilson Date February 18th, 2014

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

Question 1: What is the resolution of your implementation—that is, what is the shortest time interval it can accurately measure?

-Microseconds are the smallest accurate measurement made by the timer.

Test Plan 13-1 (Timer ADT operations)			
Test case	**Actual time period (in seconds)**	**Measured time period (in seconds)**	**Checked**
W 4	4	4.010009	
W 5	5	5.01475	
W 10	10	10.03	
W 1	1	1.00444	
W 9	9	9.03312	
W .6	.6	breaks tester	
R	1.74	1.74385	
R	5.13	5.09605	
R	8.17	8.12013	
R	9.87	9.87211	
R			

Laboratory 13: Measurement and Analysis Exercise 1

Name Duncan Wilson Date February 18th, 2014

In the table below, fill in values of *N*, 2*N*, and 4*N*: try 1000, 2000, 4000. If you do not obtain meaningful timing data—especially for binarySearch—change the value of *N* and try again.

Timings Table 13-2 (Search routines execution times)				
Routine		**Number of keys in the list (numKeys)**		
		N = 1000	2N = 2000	4N = 4000
linearSearch	O(N)	*0.569637*	1.12031	2.25228
binarySearch	O(Log N)	*0.013629*	0.014933	0.016187
STLSearch	O(N)	*0.544155*	1.08173	2.17261

Please list times in seconds

Question 1: How well do your measured times conform to the order-of-magnitude estimates given for the linearSearch and binarySearch routines?

- It matches very well. Each result pairs with the adjustment made to the number of inputs and accurately matches the prediction via big O notation.

Question 2: Using the code in the file *search.cpp* and your measured execution times as a basis, develop an order-of-magnitude estimate of the execution time of the STLSearch routine. Briefly explain your reasoning behind this estimate.

- From looking both at the code an examining a handful of results, I decided that the time complexity matches that of the linear search, O(n).

Name Duncan Wilson Date February 18th, 2014

In the table below, fill in values of *N*, 2*N*, and 4*N*: try 1000, 2000, 4000. If you do not obtain meaningful timing data—especially for quickSort—change the value of *N* and try again.

Timings Table 13-3 (Execution times of a set of sorting routines)			
Routine	**Number of keys in the list (numKeys)**		
	N = 1000	*2N = 2000*	4N = 4000
selectionSort O(N^2)	*0.521327*	*2.03895*	8.02348
quickSort O(*N* Log *N*)	*0.020409*	*0.045598*	.100581
STL sort O(N Log N)	*0.015382*	*0.034227*	.073681

Please list times in seconds

Question 1: How well do your measured times conform with the order-of-magnitude estimates given for the selectionSort and quickSort routines?

-selectionSort and quickSort match, with slight variations, to their paired big O notation defined time complexity

Question 2: Using the code in the file *sort.cpp* and your measured execution times as a basis, develop an order-of-magnitude estimate of the execution time of the STL sort routine. Briefly explain your reasoning behind this estimate.

- Due to the similarity I see in results from the STL sort algorithm and the quickSort, as well as referencing my inferences based on code examination. I evaluated that the time complexity is O(N*Log(N))

Name Duncan Wilson Date February 18th, 2014

In the table below, fill in values for the various constructor tests. Try an initial value of *N*=1000. If you do not obtain meaningful timing data, change the value of *N* and try again.

Timings Table 13-4 (Timing constructor/initialization just before vs. inside loop)		
	Constructor/initialization location	
Your value of *N*: ___1000_____	Outside loop	Inside loop
int	0.006201	0.0067635
double	0.006353	0.004168
vector	0.232595	0.449142
TestVector	0.244385	0.469616

Please list times in seconds

Question 1: For each data type, how do your measured times for the constructor just before the loop compare to the times for the constructor inside the loop? What might explain any observed differences?

- Concerning the time involved with construction of the vectors I felt that this was in good terms with my understanding of what changed. With the int and double I was time go down for the constrictor being called more than once as opposed to going up. I played with my N value and still obtained results which made little to no sense to me.