Laboratory 13: Cover Sheet

Name _Leah Kramer_____ Date _09/26/2017_____

Section _____

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 13: Implementation Testing

Name _Leah Kramer_____ Date _____

Section _____

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

Laboratory 13: Measurement and Analysis Exercise 1

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 =	2N =	4N =
`linearSearch` O(N)			
`binarySearch` O(Log N)			
`STLSearch` O( )			

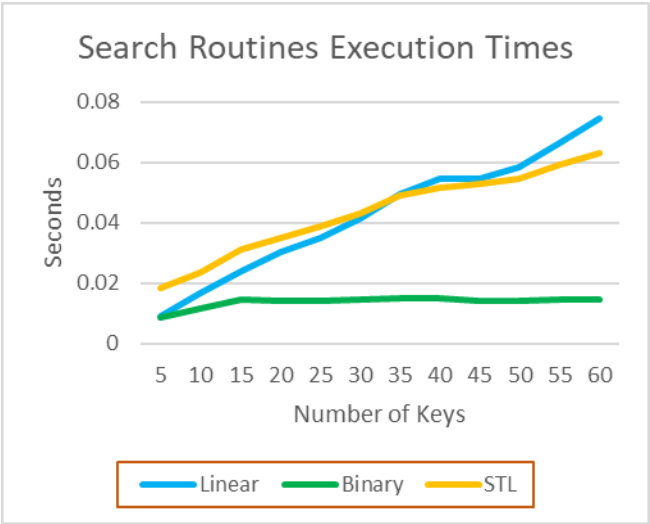
Question 1: How well do your measured times conform to the order-of-magnitude estimates given for the `linearSearch` and `binarySearch` routines?

They conform well.

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.

It would follow the same output as Linear.

	Linear	Binary	STL
5	0.008923	0.008747	0.018218
10	0.016701	0.011735	0.023633
15	0.023794	0.014341	0.031291
20	0.030489	0.014063	0.035146
25	0.035188	0.014123	0.038938
30	0.041422	0.014464	0.043195
35	0.049297	0.014996	0.04889
40	0.05457	0.014907	0.051406
45	0.054676	0.013975	0.052693
50	0.05823	0.013971	0.054629
55	0.066348	0.014427	0.059113
60	0.074557	0.014648	0.063245



Laboratory 13: Measurement and Analysis Exercise 2

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 =	*2N* =	4N =
`selectionSort` O(N^2)			
`quickSort` O(*N* Log *N*)			
STL `sort` O( )			

Please list times in seconds

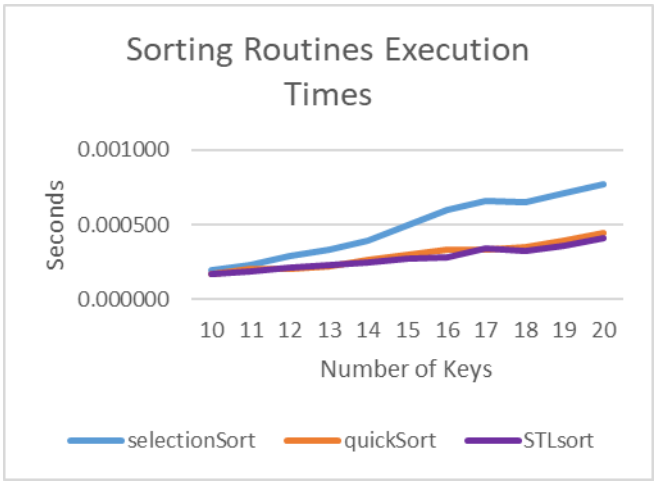
Question 1: How well do your measured times conform with the order-of-magnitude estimates given for the `selectionSort` and `quickSort` routines?

They conform well.

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.

It would follow the same output as quicksort.

	selectionSort	quickSort	STLsort
10	0.000194	0.000167	0.000173
11	0.000234	0.000205	0.000185
12	0.000290	0.000206	0.000214
13	0.000333	0.000223	0.000227
14	0.000395	0.000265	0.000247
15	0.000497	0.000300	0.000273
16	0.000597	0.000333	0.000282
17	0.000659	0.000329	0.000340
18	0.000649	0.000352	0.000328
19	0.000709	0.000389	0.000361
20	0.000774	0.000448	0.000407



Laboratory 13: Measurement and Analysis Exercise 3

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: _____
`int`
`double`
`vector`
`TestVector`
	Outside loop	Inside loop

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?

The constructor before the loop is faster. This would be because there are more actions required when the loop is called, which would require more processing time.

In the table below, fill in values for the various increment 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 13-5 (Timing pre-/post-increment operators)		
Your value of *N*: _____	pre-increment	post-increment
`int`		
`double`		
`TestVector`		

Please list times in seconds

Question 2: For each data type, how do your measured times for the pre-increment operator compare to the times for the post-increment operator? What might explain any observed differences?
I only tested for ints

	Increment	
	Pre	Post
1	0.104304	0.235906
2	0.125241	0.233968
3	0.149211	0.257165
4	0.170288	0.265644
5	0.18406	0.282574
6	0.214889	0.319981
7	0.233279	0.335108
8	0.233585	0.336728
9	0.251677	0.36277
10	0.264754	0.364613