Matthew DiGiacomo
CS260
February 6, 2015

	List (Library Data Structure)	List (ADT Array Impl.)	List(ADT Pointer Impl.)
Iterated Insertion (Front)	5.51 ms	5.19 ms	5.24 ms
Iterated Insertion (Back)	5.53 ms	5.22 ms	5.11 ms
Traversal	4.57 ms	4.17 ms	4.08 ms
Iterated Deletion (Front)	3.95 ms	3.46 ms	3.76 ms
Iterated Deletion (Back)	3.91 ms	3.25 ms	3.37 ms

	Stack (Library Data Structure)	Stack (ADT Array Impl.)	Stack (ADT Pointer Impl.)
Iterated Insertion (Push)	3.19 ms	3.22 ms	3.18 ms
Iterated Deletion (Pop)	3.12 ms	3.15 ms	3.20 ms

All timings were taken directly from the IDE I am using to write and compile my code. When the program is executed, it gives the time taken to run the main class (compile time is not included). Each value was taken 5 times and the average is shown in the table above.

Looking at the list operations performed above, it seems as though my Array and Pointer ADT implementations were fairly close in performance with regard to the timings. They do however seem to outperform the C++ built in libraries by a substantial amount in some cases. Most notably, for iterated insertion in the back, the built in function is 0.50ms and 0.59ms slower than my array and pointer implementation, respectively. Similarly for iterated deletion, the built in function is 0.66 ms and 0.54 ms slower.

Moving on to the stack operations performed above; there isn't much of a difference between my implementations and the build in functions. Across all experiments, there is never a difference between the built-ins and my implementations of greater than 0.08ms. This must mean that the C++ built in libraries are more efficient than the array libraries, or my code is just not as efficient as the code I wrote for the array implementations.

Only thing not working correctly is for iterated insertion on my Array ADT implementation. It inserts and prints out the array correctly, but the program crashes (when you start inserting more than 3 values) and I cannot figure out the cause. But it does insert the values into the array at the front and back.