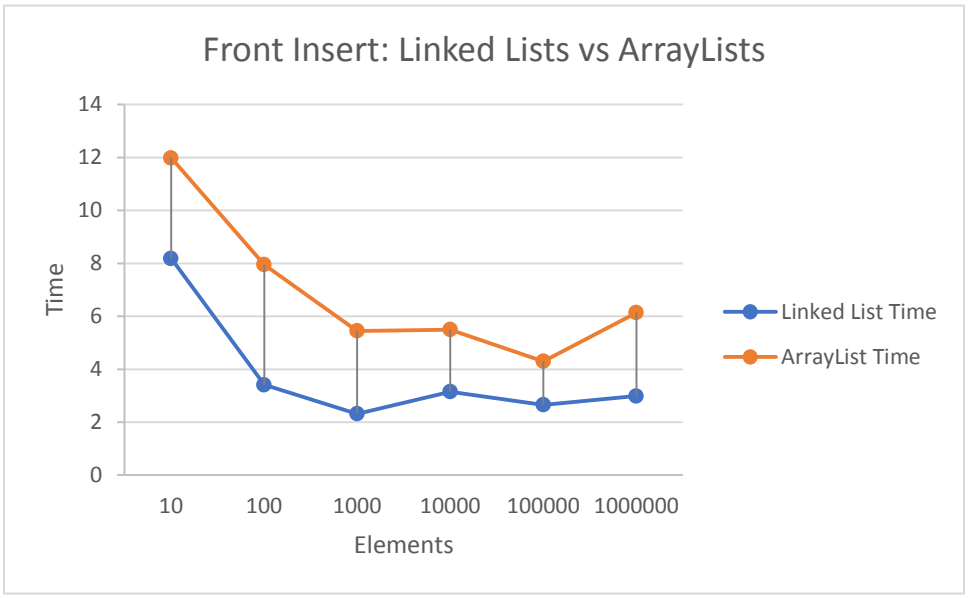
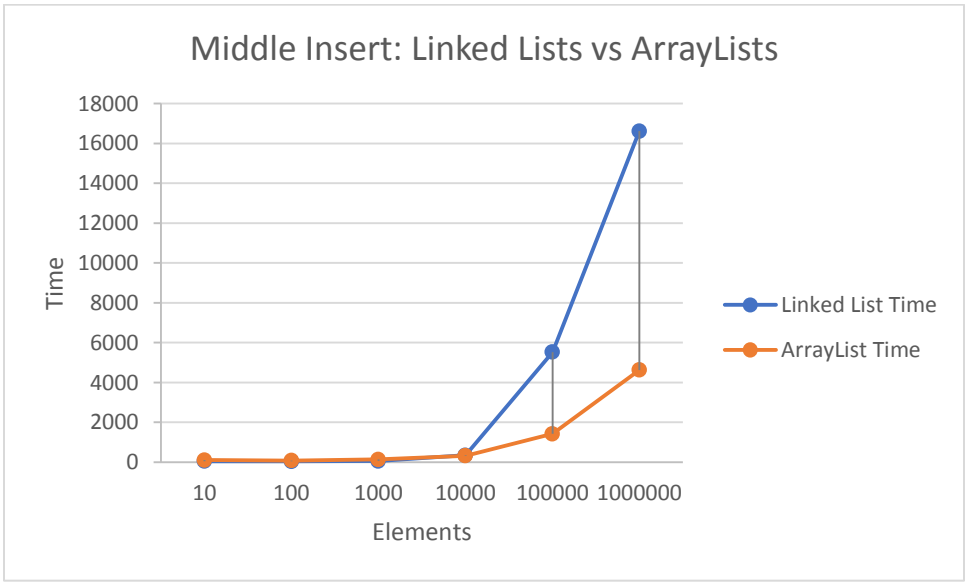
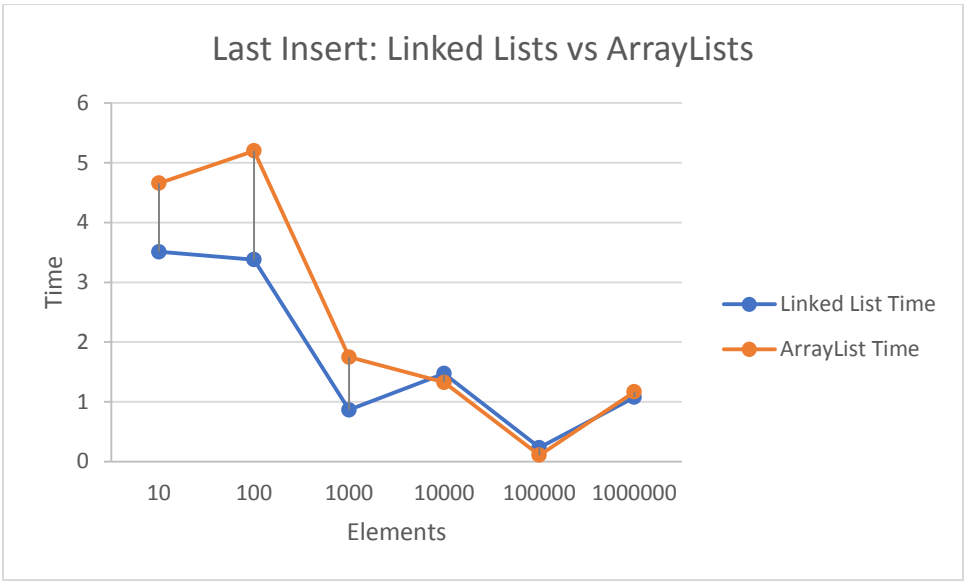
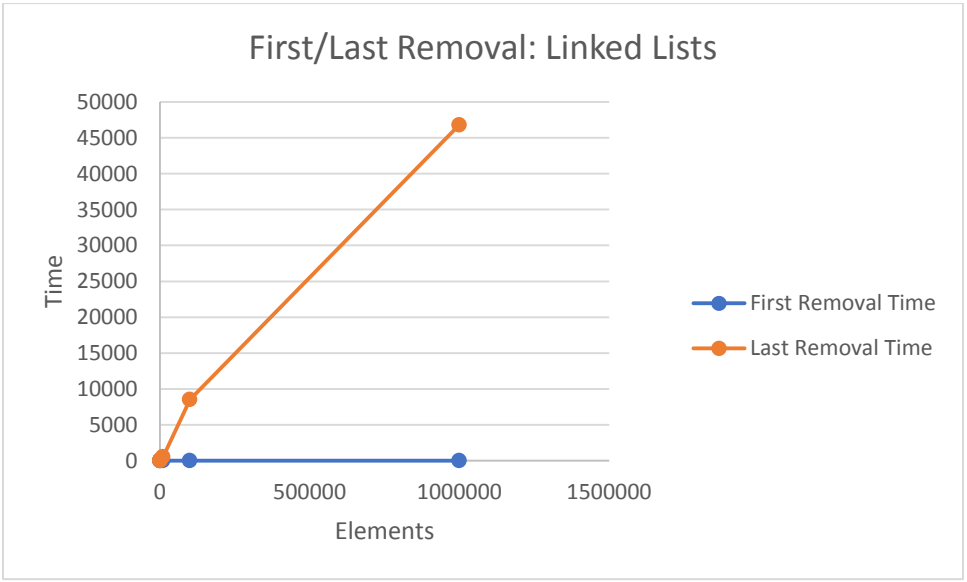
Brian Burton
CS 2420
02/24/17

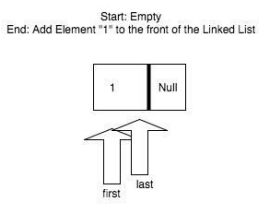
Homework 6 Analysis Document

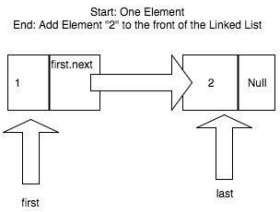


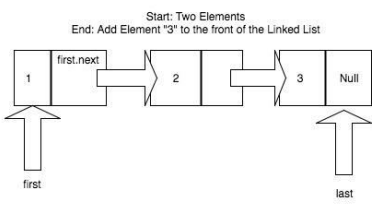


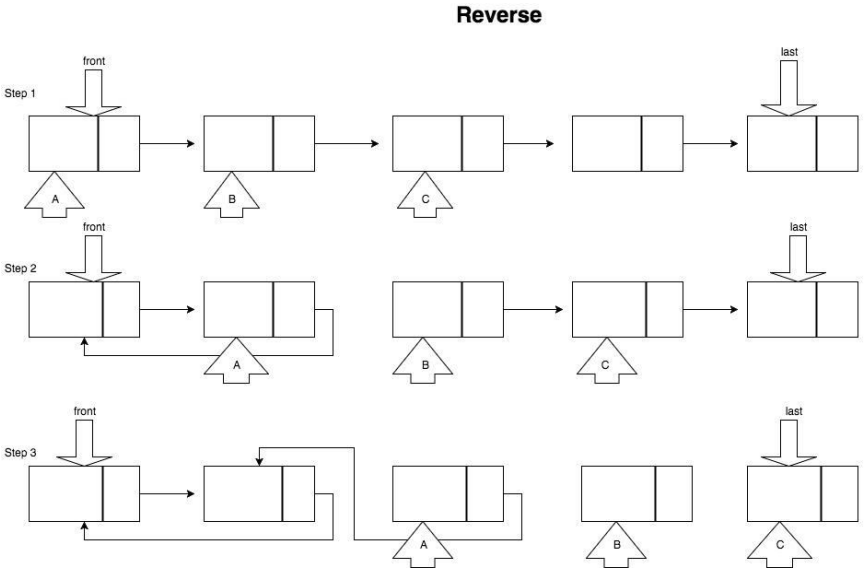

Last Insert: Linked Lists vs ArrayLists
Time
Elements
0
1
2
3
4
5
6
10
100
1000
10000
100000
1000000
Linked List Time
ArrayList Time


First/Last Removal: Linked Lists
Time
Elements
0
5000
10000
15000
20000
25000
30000
35000
40000
45000
50000
0
500000
1000000
1500000
First Removal Time
Last Removal Time









1. What the Graphs tell us

   From the graphs that I have put together, I found that in basically every case, the Linked List was faster than that ArrayList. It was faster at adding in the first position, the middle position, and the last position. My graphs also show that when adding the first and last positions, it takes longer to add less elements. This makes no sense to me. My guess is that my partner and I implemented the timing code incorrectly. The middle position graph makes more sense to me. When you are adding a few elements, it takes a short amount of time, when you try to add a lot of elements, then the time increases dramatically. This is the case because in order to add an element to the middle, you have to move every pointer that comes after that element that you added. This takes time.

   My graphs that show removing from the front and back also make sense to me. The reason for this is that it takes significantly more time to remove from the end of the array because when you remove from the front, all you have to do is move the front pointer to the next node. Removing the last on the other had is a lot harder because moving backwards in one linked lists forces you to go through ever node in the list. If we had a doubly linked list on the other hand then I would expect the two methods to have similar timing the entire time.

2. The Class Hierarchy

   The classes that I implemented interact with each other because a Linked Lists is a list of nodes that are connected by pointers. Therefore you need a node class in order to have a Linked List class. The node class supports the linked list class by giving it essential information. That information is what is in the node. What data the node has and what comes after that node.

   I am not entirely sure why the node class is necessary. I believe that the linked list could contain all of the node information by itself. However, doing this would make the readability of the code much worse. It would also make the method code much longer. And no one likes to read a lot of code.

   The node class is static class because it allows the programmer to better access the information in the node class without changing the information in the node class. The node class is also an inner class of the linked list class. This means that instead of having two java classes, we only need one. Both the node and linked list class are contained in one java class file.

3. Iteration vs Recursion

   The interactive solution to find the size of the list was a lot easier for me to implement. And it made a lot more sense to me. Whenever you add a node, add one to the size. Whenever you remove a node, subtract one from the size. The recursive solution however took a lot more thought and help. It was just hard to see how to stop the recursion so that it did not turn into an eternal loop. This of course was a challenge for all of my other recursive methods. It was also hard because if you created a new variable in the method and initialized it, when the recursion hit, it would start that variable out at the same value when I didn't want it to be. I am afraid that I still do not have a good understanding of recursion and all of it's magic. But I am defiantly getting closer.

4. Software Development Log

   I spend a lot of time on this project. Defiantly at least 16 hours but that is normal for me. The most time consuming part of the programming for me was trying to figure out the recursion and the methods that had to do with the doing something with the last position. I believe that I cannot do anything better in the future. I feel like I am doing my best. If I could take more time off of work then I would defiantly be able to do better.

   I am always one that would always like more planning. I like planning. The one thing that I wish I would have done better for this week was start earlier. I was only able to start my assignment on Monday instead of Saturday.

Regression testing is using the same variables and just changing them throughout the test so that the variables are in the state that you want them to be. I think writing the test first did help my understanding of what I was to do with my program. The problem was I wasn't getting the error messages that I am still relying on. This means that I had to write my test cases twice since they were not doing what I wanted them to be doing once I implemented the methods.

5. Thought Problems

   The reason that a singly linked list is not as good as a Queue is because it takes less steps to add to the back of a Queue than it does for a singly linked list. What could be done to fix this is to make that singly linked list a doubly linked list.

6. Finish up

   Finally, I actually really enjoyed this assignment. Probably because I understood it better than any of the other assignments that we have done so far in this class. It was also nice to be able to have pictures to help me visualize what was to happen. If I had more time, I defiantly see us being able to finish and better test our code.