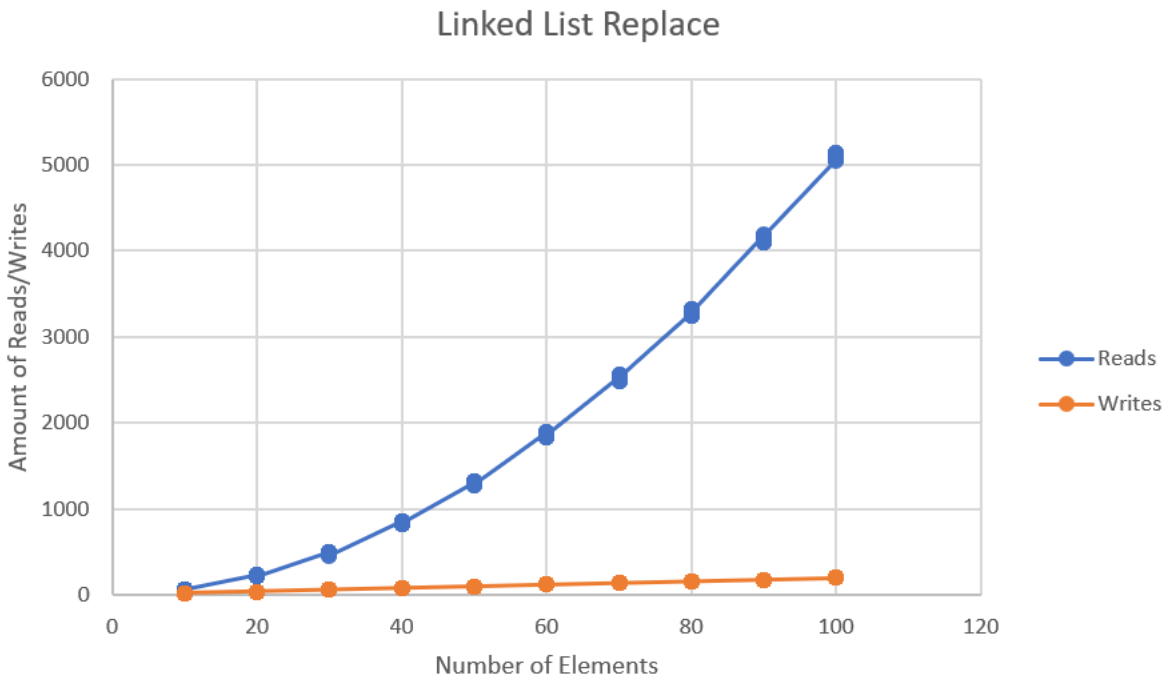
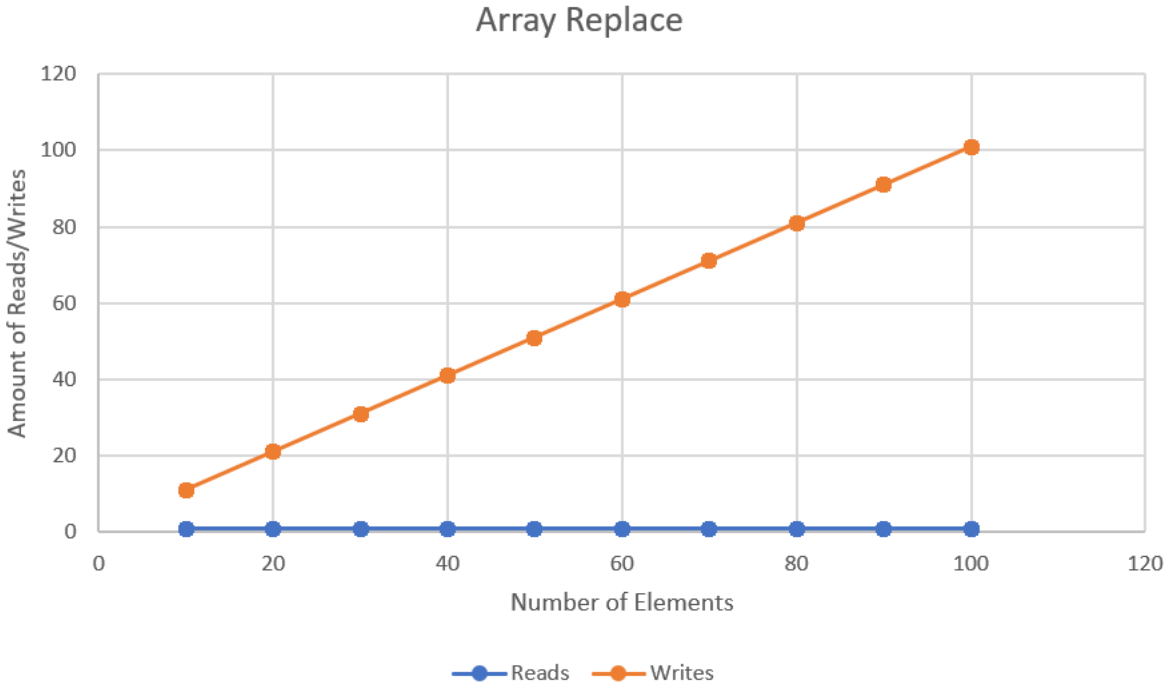
Linked Lists vs. Arrays

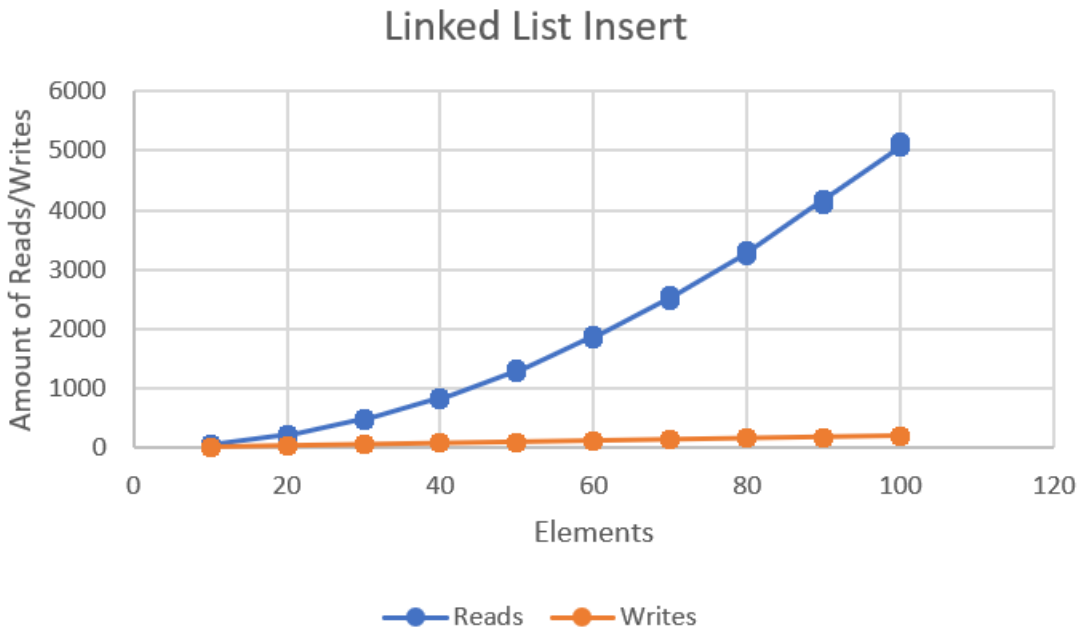
Name: Alexander Chin

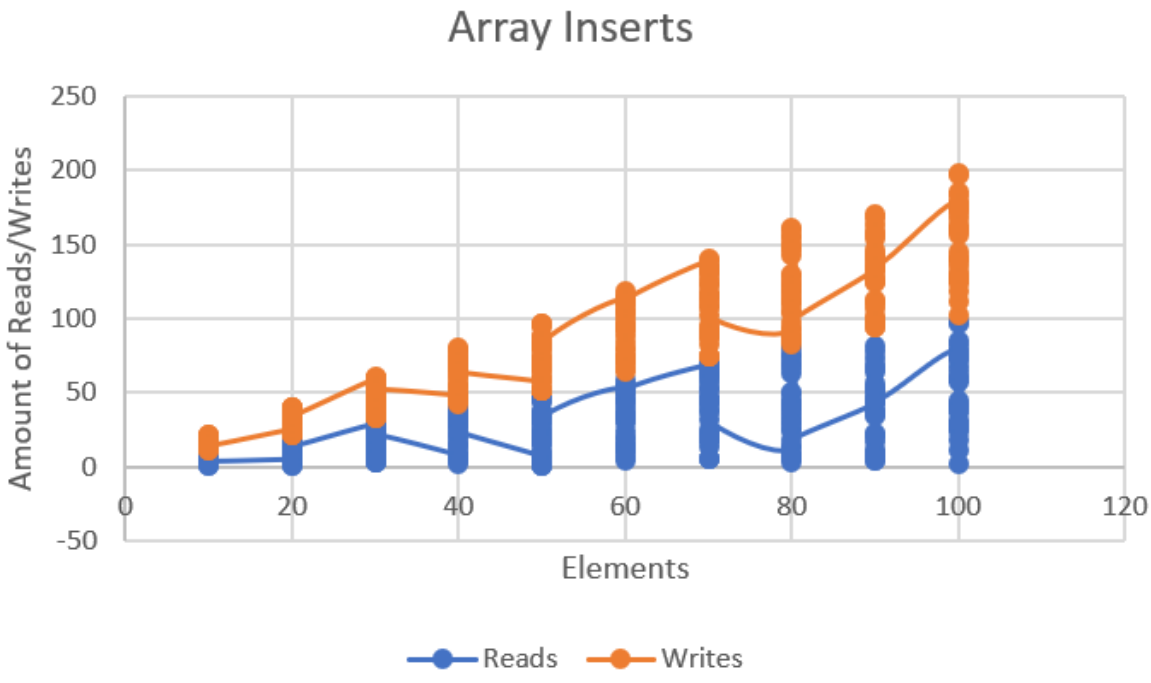
Student no. 1046121

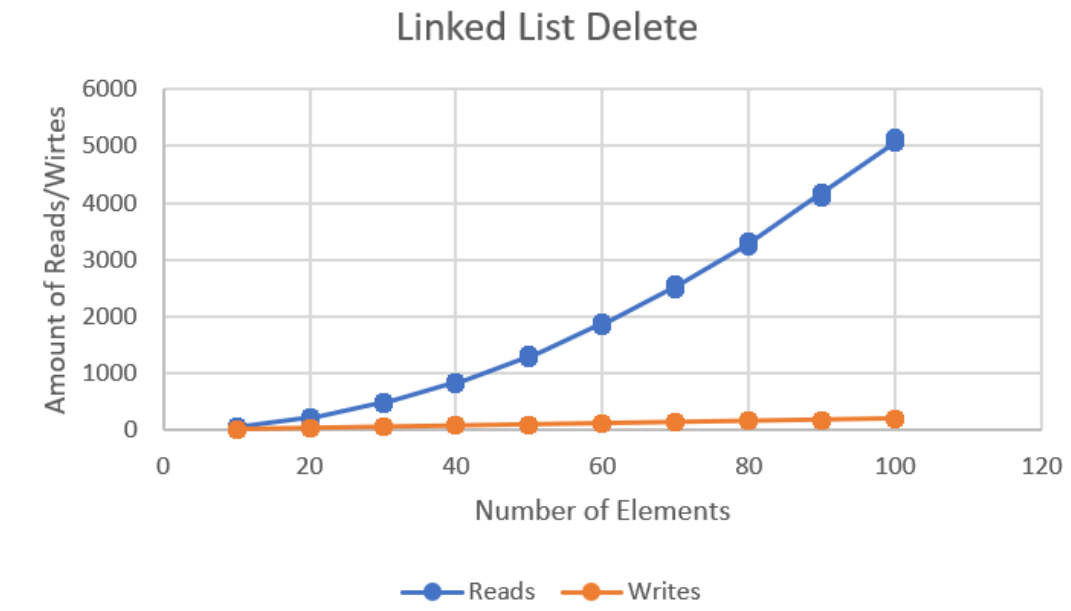
achin@uoguelph.ca

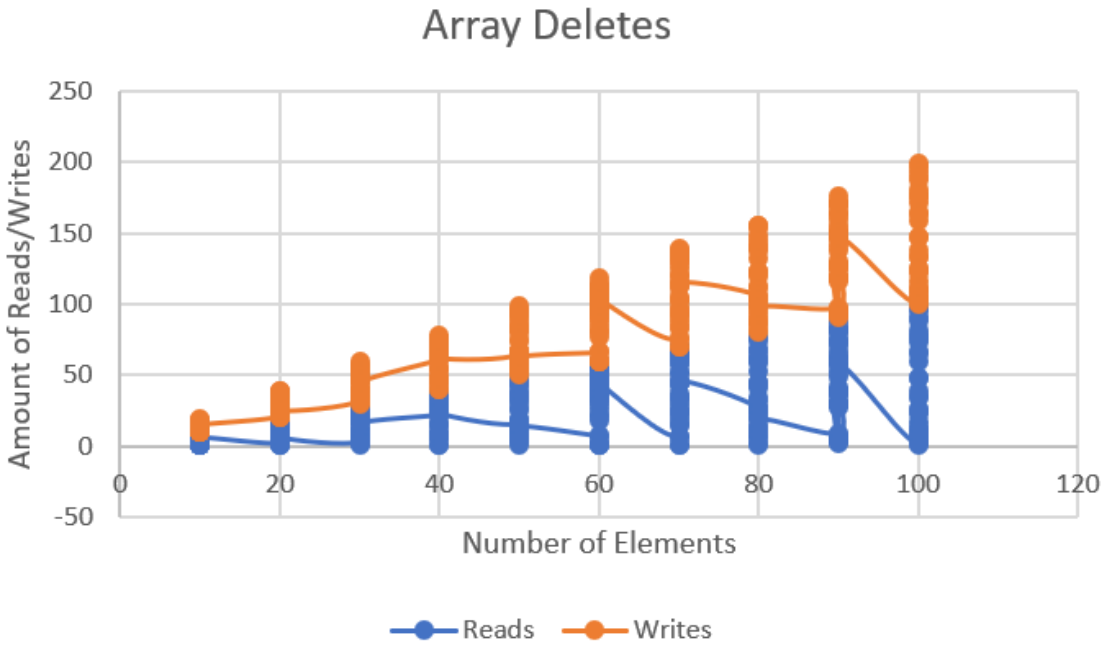


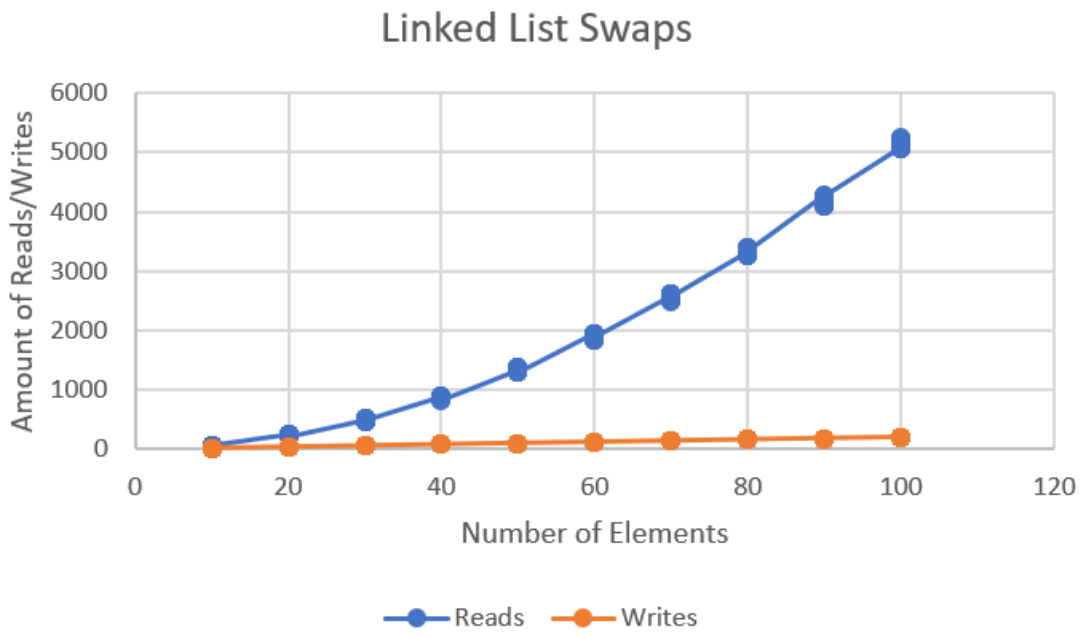


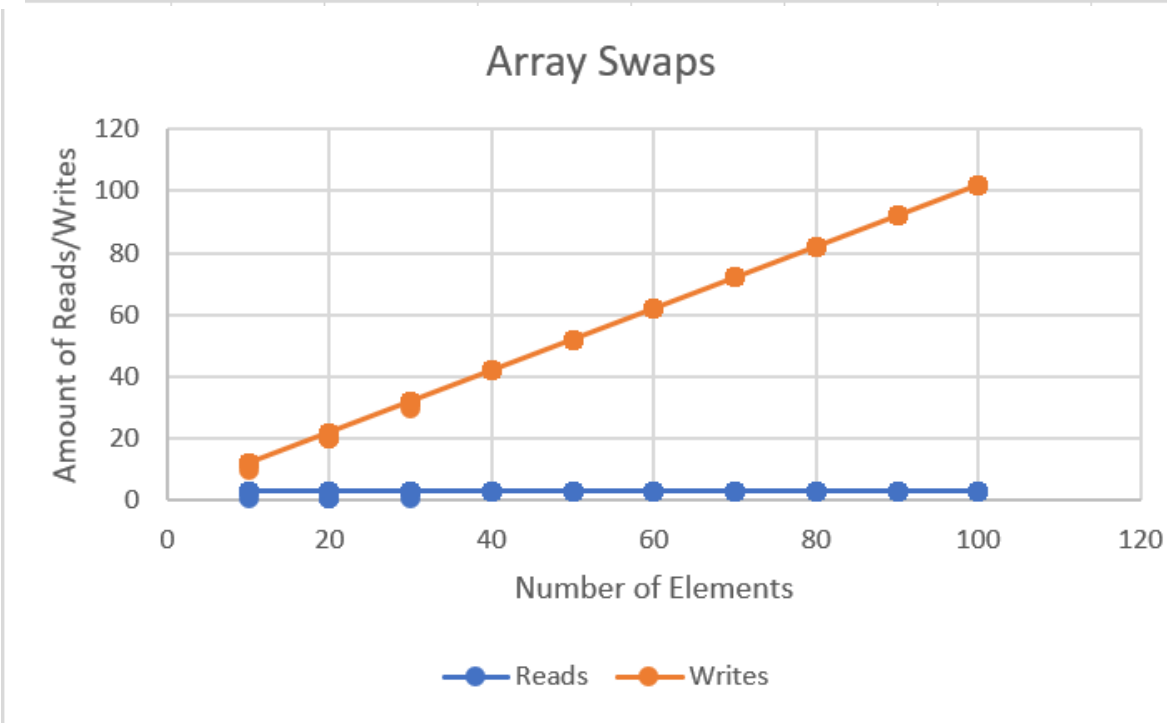

Linked List Insert
Amount of Reads/Writes
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
120
Elements
Reads
Writes


Array Inserts
Amount of Reads/Writes
250
200
150
100
50
0
-50
0
20
40
60
80
100
120
Elements
Reads
Writes


Linked List Delete
Amount of Reads/Wirtes
6000
5000
4000
3000
2000
1000
0
0
20
40
60
80
100
120
Number of Elements
Reads
Writes


Array Deletes
Amount of Reads/Writes
250
200
150
100
50
0
-50
0
20
40
60
80
100
120
Number of Elements
Reads
Writes





Discussion

Based off the above results, the advantage that arrays have over lists are that addresses are very easy to access, so operations that involve changing data like replacing or swapping are very simple. Lists have a harder time with these operations because they require moving from

the from the head to the point of interest. On the other hand, lists have the advantage when it comes to operations that affect the size of the structure, like in the case of deletion and insertion. This is because their size is fluid, and although they still face the problem of traversing from the head, they are still superior since the nodes remain at the same location. Arrays struggle here since their fixed size requires the movement of ALL elements before or after a certain point.

In general, the linked list has a consistent runtime complexity relative to the number of nodes, while the array is extremely efficient at changing certain elements due to their indices. At the same time deletion and insertion have a very good best case and very bad worse case runtimes due to the shifting of elements.