/*
* NAME: Megan Chu
* ID: A12814536
* LOGIN: cs12waot
*/

1. For Linked list:
Requires looping through all elements of the list and switching the pointer to being from the next node to the previous node. Since there are n elements in the list, this takes (big Omega)(n) times to reverse the list.

For sorted ArrayList:
Requires switching the element in the first half with its corresponding element in the back half of the list. In this case we only have to loop through half of the array list which is (big Omega)(n/2) = (big Omega)(n).

2. For Linked list:
Requires looping through all elements of the list to access current tail of the list, which takes running time (big Omega)(n).

For sorted ArrayList:
Takes one command to add to end of the list, and no looping because list does not need to be shifted, this takes running time of (big Omega)(1).

3. For Linked list:
Requires looping through list until given index to remove it. In the best case, running time is (big Omega)(1) if the index is the first node. In the worst case, running time is (big Omega)(n) if the index is the last node.

For sorted ArrayList:
Requires looping through array from point after index to shift values over one space. In the best case, the running time is (big Omega)(1) if the index is the last one in the array which requires no shifting. In the worst case, running time is (big Omega)(n) if the index is the first index of the array.

4. For Linked list:
Requires no looping, and only command to set head to second node, and remove pointer from first node. This running time is (big Omega)(1).

For sorted ArrayList:
Requires looping through all items in the list to shift them over one space, running time for this is the length of the list, (big Omega)(n).

5. For Linked list:
Requires looping through the list until some value v is found. In best case, v is contained in first node, and running time is (big Omega)(1). In the worst case, v is contained in the last node, and running time is (big Omega)(n), the length of the list.

For sorted ArrayList:
In the best case, some value v could be the first item in the list, so runtime would be (big Omega)(1). In other cases we can use a binary search since list is sorted. Runtime for binary search is (big Omega)(logn).