1. Running time for Sorted Singly LL to copy list: Theta(1); because...

Running time to copy Sorted Array List: Theta(n); because we have to create a new array List and using a for loop copy each element in the old sorted array list to the new array list in order.

2. Adding a value to the middle of the list. (where you do not want to destroy the overall order of the elements)

Running time for Sorted Singly LL to add value to the middle of the list: Theta[(n/2)]; because LL has to enter from either the head or end of the List to first point the new middle node's next node to the next node and then iterate again to the node prior to the new middle node and point the node's next reference node to the new middle node.

Running time for Sorted Array List to add value to the middle of the list: theta(n+1); because Array List has to be resorted and since memory is contiguous, it is easier to create a new Array List and instead all the values again.

3. Removing the value from the list at a given index. (where you do not want to destroy the order of the elements.

Running time for Sorted Singly Linked List to remove value at a given index is: theta(n-1) because under worst case, LL needs to iterate to the indexed node before the last index to remove the value by pointing the next reference node of the node before the indexed node to the last index.

Running time for Sorted ArrayList to remove a value at a given index is theta(n); because since Array is contiguous and sorted, ArrayList can remove the index without searching but has to reinsert all the elements by shifting the elements after the removed element(worse case, 1st element removed) to the left.

4. Removing the first value from the list.

Worst case running time for Sorted Singly Linked List to remove the first value from the list: theta(1) because header can just point to the next node to dereference the first node thus removing the first value from the list.

Worst case running time for Sorted ArrayList to remove the first value from the list: theta(n-1) because since the memory is contiguous, we have to remove the first value by shifting all the values(excluding the 1st value), 1 index to the left.

5. Determining whether the list contains some value v.

Worst case running time for Sorted Singly Linked List to know if it contains some value v: theta(n-1) because the worst possibility would be at index n-2 to get the element before the last
Worst case running time for Sorted ArrayList to know if it contains some value v: theta(1); because since it is sorted, ArrayList can just access the value at the supposed index.