For our implementation of a sorted list, we used a linked list. To handle changes to the list while iterating, we keep track of the number changes made to the list, and store that value when we first create our iterator as well. Each time the user requests the next item from the iterator, it first checks if its change count matches that of the list. If not, the iterator repositions its pointer to the next value in the list that is less than the last one that it had responded with.

Efficiency Analysis

SLCreate:
Memory: This function allocates memory that is one more than the size of the SortedList struct, which is constant. O(1)
Runtime: This function allocates memory for the SortedList and sets some default values. It has a constant runtime. O(1)
SLDestroy:
Memory: This function frees memory, and uses a constant amount in the stack for variables. O(1)
Runtime: Let n be the number of elements in the SortedList. This function iterates through each Node struct in the list and frees it, resulting in n operations. It also runs free on the list pointer itself, for a total of n+1 calls to free. The rest of the function runs in constant time. The overall runtime is linear. O(n)
SLInsert:
Memory: This function allocates memory of the size of a Node struct. This is constant. O(1)
Runtime: We insert in sorted order for this function. In the worst case, we must iterate over all n elements presently in the list. This will take linear time. O(n)
SLRemove:
Memory: This function frees memory and uses a constant amount in the stack. O(1)
Runtime: We must first find the object in the list before we can remove it. In the worst case, it isn't there and we must iterate over all n items in the list, taking linear time. O(n)
SLCreateIterator:
Memory: This allocates memory of the size of the SortedListIterator struct. This is constant. O(1)
Runtime: This function makes a single call to alloc and does a constant number of assignments. O(1)
SLDestroyIterator:
Memory: This function calls free and returns. No variables are created.O(1)
Runtime: This function makes a single call to free and returns. O(1)
SLNextItem:
Memory: This function makes a constant number of assignments. O(1)
Runtime: This function runs a constant number of assignments and then returns. O(1)
reposition:
Memory: This function makes a constant number of assignments. O(1)
Runtime: This helper function repositions the pointer of the iterator if changes have been made to the list. In the worst case, it must iterate throughout all n elements of the list. This is linear time. O(n)