Sorted List

Program Description:

- Implements a sorted list of opaque objects.

Program Functions Analysis:

- SLCreate – Allocates memory for a new sorted-list. Takes O(1) time.
- SLDestroy – Traverses through the list and deletes all dynamically allocated nodes. The running time of this function is O(n).
- SLInsert – Traverses the list and locates its sorted position to be inserted. If the item is not a duplicate, it's inserted into the list. Duplicate items are freed and an error message is printed. The running time of this function is O(n).
- SLRemove – Traverses the list and compares the given object to the objects in each node. If a match is found, the function checks if there are any iterators pointing to that node. If there are no iterators pointing to the node, then it is simply taken of the list and the node is freed. If the node has an iterator pointing to it, the isValid flag is set to false, this is used during the SLNextItem function to not return any items that are still in the list but were "deleted". Lastly, it also behaves as a sort of garbage collection. While looking for the newObj to be deleted, it also looks for any items that are no longer valid, and that no longer have pointers pointing at them, and free them. The running time of this function is O(n).
- SLCreateIterator – Allocates memory for a new iterator, points the iterator at the front of the list and increments the pointer count of the front node. If an iterator is created for an empty list, an error message is printed to the screen. The running time of this function is O(1).
- SLDestroyIterator – Decrements that number of pointers of the item it is currently pointing to and releases the memory that was allocated for the given iterator. The running time of this function is O(1).

- SLNextItem – Checks if the node it’s currently pointing to has been removed (has a false isValid tag). If it hasn’t been removed while the current pointer pointed to it, the node’s data is stored and then iterates to the next node. If the node was removed while the current iterator pointer pointed to it, it decrements the pointer count of the current node and if the next node isn’t null, it increments the pointer count of that node. The running time of this function is O(1).