readme.pdf for the sorted-list data structure

The following is a brief description, runtime analysis and memory usage of each function contained in the sorted-list.c data structure:

1. **SLCreate**: Creates a SortedList given a CompareFuncT pointer and a DestructFuncT pointer.
   Runtime analysis: O(1)
   Memory usage: Allocates 24 bytes of memory on the heap for the SortedList data structure. Contains 1 local variable of size 8 bytes on the stack.

2. **SLDestroy**: Destroys a SortedList by first destroying all nodes that are not being used by iterators and all data contained therein. Then the list itself is destroyed.
   Runtime analysis: O(N)
   Memory Usage: Deallocates 24 bytes of memory on the heap for every ListNode that is freed. Also, deallocates the data contained within the nodes for every node freed (amount of memory freed dependent on size of data item within the node). Then, deallocates 24 more bytes when the SortedListPtr itself is freed. Contains 3 local variables each of size 8 bytes on the stack.

3. **CreateNode**: Creates a new ListNode struct given a new data object and returns a ListNodePtr to this new struct. *Internal function not for use by the user.
   Runtime analysis: O(1)
   Memory Usage: Allocates 24 bytes of memory onto the heap for a new ListNode struct. Contains 1 local variable of size 8 bytes on the stack.

4. **SLInsert**: Inserts a new node into the list containing the given data object in sorted (descending) order. *Given that data is valid and does not already exist.
   Runtime analysis: O(N)
   Memory usage: Calls CreateNode, which as stated above allocates 24 bytes of memory on the heap, only if a new node is successfully inserted (unsuccessful insertions are, for example, duplicate items). Contains 3 local variables each of size 8 bytes on the stack.

5. **SLRemove**: Removes a node from the list containing the given data object. If the data is not found 0 is returned to signal a failed removal.
   Runtime analysis: O(N)
   Memory usage: If an item for removal is found, deallocates 24 bytes of memory from the heap that was previously used for the ListNode containing the data item and also deallocates the data contained within that ListNode (the amount freed dependent on the size of the data object itself). Otherwise, no memory is freed. Contains 4 local variables each of size 8 on the stack.

6. **SLCreateIterator**: Creates a new SortedListIteratorPtr which can be used to iterate through the given SortedList.
   Runtime analysis: O(1)
   Memory usage: Allocates 32 bytes of memory on the heap for a new SortedListIterator struct. Contains 1 local variable of size 8 bytes on the stack.

7. **SLDestroyIterator**: Destroys all nodes where the iterator is the only remaining pointer to that node and the data contained therein. Then the iterator itself is destroyed.

Runtime analysis: O(N)

Memory Usage: Deallocates 24 bytes of memory from the heap for every list node that is destroyed. Also deallocates an unspecified amount of memory from the heap for every data item that is contained within the destroyed ListNodes (amount of memory deallocated dependent on size of data item stored within the node). Finally, deallocates 32 bytes of memory from the heap that was previously used for the now destroyed SortedListIterator struct. Contains 2 local variables each of size 8 on the stack.

8. **SLNextItem**: Returns a void pointer to the data of the next item in the list. (If a data item is removed while an iterator is still "on" it, the iterator can still get data from this node and iterate to the next item in the sorted list as usual). May destroy node and data if after iterating over a node, there are no more pointers left on it (usually occurs when nodes are removed from the sorted list but remain "stuck" to iterators).

Runtime analysis: O(1)

Memory usage: If it is determined that a node needs to be removed, deallocates 24 bytes of memory from the heap that was previously being used by the removed ListNode struct and also deallocates an unspecified amount of memory from the heap for the data item contained within each removed node (amount of memory deallocated dependent on the size of the data item) (may remove multiple nodes). Contains 2 local variables each of size 8 bytes on the stack.

9. **SLGetItem**: Returns the data object as a void pointer of the last item returned by SLNextItem.

Runtime analysis: O(1)

Memory usage: None (no local variables and no (de)allocations to or from the heap).