Zachary Scott

ASSIGNMENT 2 README

Big-O Analysis:

SLCreate, SLCreateIterator, SLDestroyIterator all have O(1) for memory and runtime. They are simply “constructors” as they would be referred to in OOL’s, and generate space for a given struct.

SLInsert places the new Node at the front of the list and bubbles the data through the list to the correct position. It does this by continually swapping data between the pairs until the new data is in the right place. This is easier than moving the “nexts” of every node and is a cleaner implementation. Running-Time: O(n) worst case, O(1) Memory.

SLDestroy deletes nodes one at a time, so running time is always O(n), O(1) Memory.

SLRemove has O(n) running time worst case. (If the target data is at the end of the list). O(1) memory.

SLNextItem and **SLGetItem** both have O(1) for running time and memory. They simply return the data from a given node/iterate a pointer.

Other Notes:

The “ghost” flag (*Halloween season baby*) in the Node struct refers to whether a Node has been removed from the list, but shouldn’t be freed from memory due to existing references. (Hence the output “A Node has been ghosted”).

SLGetItem returns the data from the iterators current node, but does not iterate.
SLNextItem moves the iterators “current” pointer to the next node, and returns that data.