Ronald Salinas CS162

ASSIGNMENT 5: Linked Lists

Design

Initially, I had planned on having the list able to be traversed back and forth, as a sort of bidirectional link list. But that as though it would stray too far from the assignment design.

For every single function which modifies the linked list, I had to include several conditional statements during implementation for several different cases of the link list (if it was empty, if there was only one node, etc.).

The “insert_middle” function had originally been designed to only be used if and only if there were already at least two pre-existing nodes in the linked list. During implementation, it had been changed to insert at any arbitrary spot in the list.

I had wanted to do a swap-by-node method for the sorting functions, but I copped out and did a swap-by-value.

Testing

Prompt	Input Values	Expected Output	Expected?
(1) Push or (2) Append	1	Normal execution of push function	Yes
	2	Normal execution of append function	Yes
	3	Error as 3 isn't an option	Yes
Enter number:	1	Normal push/append of 1	Yes
	0	Normal push/append of 0	Yes
	d	Type mismatch since d isn't a number	No (Returned an address)
	007	Error since value contains leading 0's	No (Ignored 0's)
Insert numbers? (y or n)	y	Normal execution of insert function	Yes
	n	Skip insert function	Yes
	d	Skip insert function	Yes (Only accepts 'y')

Insert where? (index)	0	Inserts at beginning of list (push)	Yes
	Something between list	Inserts in the middle of list (insert)	Yes
	End/beyond list	Inserts at end of list (append)	Yes
Remove numbers? (y or n)	y	Runs remove function	Yes
	n	Does nothing	Yes
	asdasdasd	Does nothing	Yes
What to remove?	Value in the list	Removes all nodes of that value	No (Only removes one at a time)
	Remove a value repeated at beginning	Removes all nodes of that value	No (Only removes one at a time)
	Remove a value at end (only two nodes)	Removes all nodes of that value	No (Returns address)
	Remove a single node (list only has one node)	Removes that given node	Yes
Sort ascending, descending, or neither:	a	Sorts list by ascending values	Yes
	d	Sorts list by descending values	Yes
	anything that is not a or d	Leaves the list as-is	Yes
Do this again? (y or n)	y	Runs clear function and repeats process	Yes
	n	Exits normally	Yes
	Anything not y	Exits normally	Yes