Emmanuel Lennix

Test Cases:

* A_0 = Before
A_1 = After

Function	Description	Example	Pass/fail
insertAtPosition (insert 2)	insert at beginning	A_0: 10→5→6→10 A_1: 2→10→5→6→10	pass
	insert at any position between first and last	A_0: 10→5→6→10 A_1: 10→5→2→6→10	pass
	insert at last position	A_0: 10→5→6→10 A_1: 10→5→6→2→10	pass
	insert at position thats greater than length	A_0: 10→5→6→10 A_1: 10→5→6→10	pass
	If list is NULL	A_0: NULL A_1: NULL	pass
	If list is empty	A_0: {} A_1: garbge returned	pass
deleteOccurrences (delete 10)	At beginning, end or middle	A_0: 10→5→10→6→10 A_1: 5→6	pass
	If not in list	A_0: 5→7→9→12 A_1: 5→7→9→12	pass
	list is null returns NULL		pass

Test cases

Emmanuel Lennix

function	Description	Example	Pass/fail
MoveMaxAtEnd	If max at first, end, or in between first and last	A_0: 10→10→5→6→10 A_1: 5→6→10→10→10	pass
	If list is empty	A_0: { } A_1: move biggest garbage value to end	pass
	If list is NULL	A_0: NULL A_1: NULL	pass
Sublist	|A| > |pos_list|	A_0: 10→5→6→10 pos_list_0: 0→3 A[pos_list]: 10→10	pass
	|A| < |pos_list|	A: 10→5→6 pos_list: 1→0→2→1 A[pos_list]: 5→10→6→5	pass
	If position doesnt exist in A	A: 10→5→6→10 pos_list: 0→8→1 A[pos_list]: 10→5	pass
	If A is null	A: NULL Pos_list: 0→1 A[pos_list]: NULL	pass

Emmanuel Lennix

Task 2

Time Complexity

deleteOccurrences()

- this function would only loop through N times, therefore it's time complexity is $O(N)$

Sublist()

- Because this funtion loops through to N and has a nested recurssive function called find within it that also runs to N. The time complerity is $O(N^2)$

insertAtPosition()

- this function is broken into two parts. If you want to insert at start it is constant time, but in most other case it will typically be $O(N)$

because it will loop thrugh until it reaches the N^{th} position and break.

Emmanuel Lennix

Task 2

moveAllMaxAtEnd()

- This function doesn't have any nested loops, but it contains 1 while loop that runs N, and a for loop that runs to C. Plus 2 calls to reverse and 1 call to delete which both run to N giving us the equation.

$$N + 2N + N + C$$
$$4N + C$$

∴ time complexity is

$$O(N)$$