Building a Linked List

Linked lists provide a flexible data structure with new algorithmic strategies for implementing a list. This exercise is a mirror of the Array List exercise; only the implementation should be entirely a linked list. The book provides specific guidance for the algorithms for implementing a linked list, beyond that in the videos, so use that as support. For this implementation, your list should be iterative, *not recursive*. You should use looping to realize your algorithm. We will build the linked list again (remember that practice helps to cement ideas in our brain!) after we discuss recursion.

"C" Grade

For the basic assignment, build the following functionalities using the starter code provided:

- size – returns the size of the list
- isEmpty – return true if the list is empty
- add – Add an item to the end of a list (returns true if added)
- remove – removes an item at the specified index from the list and returns that item (or a null value if the index is invalid)
- clear – clears all values from the list
- get – returns the item at the specified index (or a null value if the index is invalid)
- set – changes the value at the specified index and returns the value that was replaced (or a null value if the index is invalid)

"B" Grade

For those who want credit towards a "B", you should also implement the following

- contains – returns true if the list contains the provided value
- indexOf – returns the index of the first occurrence of an object in the list
- lastIndexOf – returns the index of the last occurrence of an object in the list
- remove – removes the first occurrence of an object in the list and returns true if a value was removed or false if the value was not present

"A" Grade

Finally, for those who seek to earn an "A", you will implement a slightly unusual additional list implementation (do not use the same code like the one above!). Your list should also act as a set. Remember that a set only accepts a single copy of the contained data. If I add "a", "b", "a" and "b" to a set, the set will contain only 2 items, not the 4 that a list would. Your "Array Set" will have all the features above, except that it will not allow for more than one copy of a value to exist in the list. Your changes will primarily be in the add and set functions. You must implement this list separately, as it will not pass many of the test cases. To complete these last tasks, you need to:

- Copy the "B" level code and then add the logic to make your new list behave like a set
- For each failed test, explain why the test case is now in error rather than your code. If multiple tests fail for the same reason, you can condense your explanations, but makes sure you address each test failure by showing the printout from the ListTester.
- Finally, discuss the reasons that an Array may not be the best data structure for managing a set to the best of your knowledge at this point. You do not need to read ahead or do extensive research. I merely want you to reflect on what you know about lists and sets and arrays at this point and consider any faults or advantages in using arrays and lists in conjunction with sets.