Project 2: Literally Loving Linked Lists LOL

Test Cases:

To test my linked list, I created various test cases that goes through every method of the code. I made a test case with an empty list, with one element, and one with multiple elements. As expected, when I went through the test cases, I identified some bugs in my code and had to change it accordingly so that it is compatible with an empty list and a list with only one element; and hence, by using these test codes, I was able to improve my linked list. Nevertheless, during my final test run, I was able to ensure that my code is compatible with all scenarios possible. To begin with, I went through the code using valid values to make sure the code responds as I wanted. I then proceeded to test invalid values using the try method. As the invalid values were supposed to have an IndexError, I made an exception and printed 'PASSED' when the code encountered an IndexError. By testing these three scenarios, that of an empty list, a single element list, and a multiple element list, I was able to validate my code. Furthermore, by providing invalid and valid inputs for each of these scenarios, I was able to ensure that my code works in all scenarios.

append_element has a constant-time performance characteristic – O(1). This is because the code does not iterate through the list; and hence, has a constant-time performance.

insert_element_at has a linear-time performance characteristic – O(n) with an exception of inserting an element at index 0. It has a linear-time performance as this method requires iteration through each element until it gets to its desirable location. However, if the desired location is at index 0, it has a constant time performance as it does not have to iterate through any elements.

remove_element_at has a linear-time performance characteristic – O(n). Similar to insert_element, this method has to iterate through each element to reach its desirable location.

get_element_at has linear-time performance characteristic – O(n) with an exception of inserting the element at index 0. It has a linear-time performance as this method requires iteration through each element until it gets to its desirable location. However, if the desired location is at index 0, it has a constant time performance as it does not have to iterate through any elements.

Josephus has a linear time performance characteristic – O(n). This is because it uses a for loop to create a linked list for the n number of rebels; and then uses a while loop to remove or kill the necessary rebels until there is a single survivor; and hence, Josephus has a time performance of O(n).