a. Code in python file

b. Code in python file.
    i. The Upper bound is O(nk) because the function uses array slices to check data in the array to compare to the linked list items. It has to iterate through n items for the array and the linked list separately which would be n iterations for n characters. Therefore, it would be n+n(k) which can be simplified to big O(nk). However, if you include the call of the convert_str function to change the string to a linked list, then it would be much greater.

c. In order to make this more efficient, I could just not use slices to drop the function to O(n). However, I think the most efficient way would be to just compare the original string to a reversed version of the string. Then you would not have to convert the string to a linked list or append the characters to a list in order to compare the two. Instead, you could reverse the original string and compare it to the original unreversed string. Depending on the method of reversing (For example: recursively w/ slices - O(nk) but still less than the method in part b.), the time taken should be much less than the process of converting the string to something else entirely and then comparing to check if it is a palindrome.
    Using the method of reversing the string recursively using slices is still O(nk) but it is also less than the other function which is actually O(n+nk). Especially when you think about the process of having to convert the string to a linked list first.

d. Code in python file

e. Code in python file