In the textbook, it talks about the positional list, which includes the abstract value p, and it is built based on the doubly linked list class. In the positional list, it has a special variable which is *p* , an abstract value. The most different thing between two lists is caused by the uniqueness of the position abstract value. Because of this value, we can find the element of that position in the positional list if we know the position we want to find. For the get_element, remove, insert method, we can use this position to get elements quickly by constant time. However, for Linked List, to find the value, we need to use a for loop to get to the index of the value which means we need to run the linked list to find the value, and the worst case operation time is O(n), which is linear increasing by the size of the linked list, instead of the constant time. This can benefit the whole class function.

But, if we don't know where the position is in the positional list, we still need to run the list and find it out which will be the same as the linked list. This will make the method in both positional list and linked list which need to find the index(position) value and have constant steps on every node(p) to have linear operation time O(n).

Another thing is, because the positional list has one more attribute than the linked list which is position abstract value. It leads to the problem of larger storage needs for this list than the linked list.