Question 2

Referring to the proposed implementation in Question 1, analyze the total time required to push n elements to the Stack and express it using the Big O notation.

Next, analyze the total time required to pop those n elements from the Stack amd express it using the Big O notation.

A detailed analysis is expected, i.e., if you present a final answer only, you will be unhappy with your grade. In othere words, "show your work".

Type your analysis and save it in a file called Analysis.pdf and submit to CourSys.

Response:

Every time we push an element into the list we use our tail StackNode pointer. Therefore it requires a constant amount of operations.

Ex:

```
tail->next = node;
tail = node;
```

Therefore we can do n pushes with n operations. Therefore the amount of time it takes to push n elements scales linearly. Or in Big-O notation, it is O(n).

On the other hand, because we are using a top-tail implementation of our stack, popping is much more time expensive!

It is not expensive to peek the value we are to return, that can be done in O(n) by using `tail->data`. The expensive part is because we are using a DHSL list. Which means for each pop we must traverse through the n-1 proceeding nodes so that we may get the node previous to the current tail. After that we need to "chop off" the tail from our list by deleting the current tail, setting the current tail to the prev element, and then setting that new tail's next to NULL.

So for each pop of an element we must make n-1 operations. This O(n-1) simplifies to O(n).

To pop n elements we must make n operations n times where n is decreasing by one each time. Resulting in O(n) * O(n) -> O(n*n) -> O(n^2). This is not very good!