A) The goal of the problem is to write a simple program that can read a file with integers and puts all of integer values in a sorted manner into a linkedList. Then report the minimum, maximum and median value from the linkedList. Also to report how much time it took to insert the values into the linkedList, minimum, maximum and median value.

B) To find the minimum value from the list, I took the head of the linkedList. As the linkedList is sorted from minimum to maximum. So the head of value will be the minimum given the linkedList is sorted. I did some tests while I was implementing the inserting and sorting process to make sure the linkedList was sorted correctly.

To find the maximum value I took the last link/node from the list. Given the same idea/reason as with finding the minimum.

To find the median I divided the size of the list by two to get the middle index. And checked if the size of the list was odd. If it is odd the middle index is the correct index for median value. If the size is even then we take the middle value and the value before that and find the average of the to find the median of the list.

Any other potential alternative approaches that I can come up which will result in a poor timing are:

- If we were to check each value against every value in the list to find the min and max. Which algorithmically doesn't make sense as the requirement for the problem is to create a sorted list. And it is intuitively right to use that requirement to our advantage.

C) CPU maker and model: Intel(R) Core(™) i5-7300HQ (laptop)
CPU Speed: 2.5GHZ
RAM available: 7.98GB
Hard drive type: Solid-State Drive (SSD)
O/S: Windows 10 Home
Javac version: 11.0.2

I ran the test with input1.txt for about 5 times to see if there was any significant changes in the timing. Which there wasn't any.

Cont next page…..

From input1.txt:

D) max: 4000
mins: 1
med: 2056

insert_time: 175066 microseconds
max_time: 3 microseconds
min_time: 0 microseconds
med_time: 23 microseconds

insert_time is the highest due to how many times we have to traverse through the linked list.
max_time is longer than min_time because we are traversing through the link one time

Med_time is also reasonable as there is a method call. And list.size() call happening. Then we traverse through the list to find the value or values.

My tests with input2.txt was taking so long. I at least let it run for 20 min and never got any results. I think the reason is how I'm creating the sorted list. My algorithm traverses through it multiple times and there I use get(index) methed and add(index,value) in a loop. That might be causing very poor time performance. But with the current knowledge this was all I could come up with. I've been trying to come up with multiple other ways to solve this timing problem and I couldn't figure it out.

I'm planning to discuss with classmates after we get our grades back.