Unit 7 Assessment

Step 1 Runtime Analysis

Findings over 2 iterations (each array size ran twice for each function)

Run	ArraySize	InsertRuntime (ms)	AppendRuntime (ms)	AppendFaster	Rate
1	extraLarge	1745.429	7.200	True	242.42
2	large	24.837	1.823	True	13.62
3	medium	1.053	0.513	True	2.05
4	small	0.137	0.249	False	0.55
5	tiny	0.149	0.376	False	0.40
6	extraLarge	1779.831	6.623	True	268.72
7	large	18.360	1.381	True	13.29
8	medium	0.541	0.424	True	1.27
9	small	0.173	0.340	False	0.51
10	tiny	0.162	0.377	False	0.43

The Append function clearly scales better than the Insert function.
Observations:

- At a ‘tiny’ to ‘*small’* size the **Insert** function will run measurably faster, but the difference is negligible.
- Once the array size reaches medium, the The **Append** function begins to take the lead, but the difference is still negligible
- At *‘Large’* size the difference in scale becomes clear with *‘ExtraLage’* making the advantage of the **Append** function obvious.
- The **Append** function will run about 250 times faster than the **Insert** function over an array of 100000 elements.
- At scale, the **Insert** function will begin to eat up seconds, which can be quite significant given today’s computing capabilities.

Conclusion: Append will scale at an exponential rate.

Reason: *Why is ‘Append’ better in this case?*
"The doublerInsert function requires additional actions for each element in the array as it loops through because it shifts all existing elements to as it adds the new ones at the beginning.
The doublerAppend function simply adds a new element to the end of the existing array without needing to modify the elements already present.
Consequently, doublerAppend has a time complexity of *O(n)*, which is linear, while the doublerInsert function has a time complexity of *O(n^2)*, which is exponential. The exponential

complexity occurs because, as the number of elements increases, the time required to process them increases by a factor of *n*; the Insert function requires that action is taken on all elements in the array as it loops over the array.