Brittany Hancock
IT415: Lenore Montalbano
Project 2: ArrayStringListHelpers.cs (Capstone Module)
October 5, 2025

InsertIntoArray inserts a value into the middle (or any index) of an array by shifting elements to the right. The last element is overwritten in a fixed-size array. It has a time complexity of O(n) due to the shifting operation, and a space complexity of O(1) since it modifies the array in place. The trend observed is that as the array size increases, the time required for insertion also increases linearly.

DeleteFromArray deletes an element at the given index by shifting elements left. The last element becomes 0. With a time complexity of O(n) since we may need to shift the n-1 elements, and a space complexity of O(1) since we modify the array in place without using additional data structures. The noted trend is that shifting elements in an array is inherently linear in terms of time complexity.

ConcatenateNamesNaive concatenates names using += with a space before each name. With a time complexity of O(n^2) due to string immutability. The trend noted is that using += for string concatenation in a loop is inefficient for large datasets.

The ConcatenateNamesBuilder efficiently concatenates names using a StringBuilder with spaces. With a time complexity of O(n) amortized time, it is significantly more efficient than ConcatenateNamesNaive, which has a time complexity of O(n^2) due to string immutability. The trend noted is that for small n the difference is negligible, but as n grows, the StringBuilder approach becomes significantly faster.

InsertIntoList inserts a value into a List at the given index. With the time complexity of O(n) for middle inserts (array-backed shifts), and amortized O(1) when adding to the end if capacity is available. The trend noted is that inserts at the start or middle of a large list take longer than inserts at the end.