InsertIntoArray:

* Creates and returns a new array with a specified value inserted at a given index, shifting the remaining elements one position to the right
* O(n) Two for loops run through parts of the array to copy values into the new one
* Arrays are fixed in size, so inserting requires allocating and copying — costly in both time and memory

DeleteFromArray:

* Returns a new array with the element at the specified index removed, shifting remaining elements left
* O(n) Two for loops run through parts of the array to copy values into the new one
* Deletion is expensive due to shifting and resizing

ConcatenateNamesNaive:

* Builds a single string from an array of names using +=, separated by commas
* O(n2) Since string are immutable, each concatenation copies the entire existing string to memory
* StringBuilder should always be used in place of this

ConcatenateNamesBuilder:

* Builds a string of names using a StringBuilder
* O(n) Appends with StringBuilder are each O(1)
* Much more performant than using +=

InsertIntoList:

* Inserts a value into a List<int> at the specified index, shifting subsequent elements.
* O(n) Inserting at the middle or beginning requires shifting all subsequent elements. Inserting at the end is O(1) (due to preallocated buffer space).
* Inserting at the end is more performant than inserting into arrays.