Array Manipulation and Analysis COW

Level 1

Complete the following methods in the ArrayContainer:

Name:	swap
Input:	int index1, int index2
Output:	nothing
Action:	swaps to numbers in the array located at the two indexes passed in. So, if the array stores {34, 88, 53, 97, 21} and 1 gets passed in for index1 and 3 gets passed in for index2 then the array should store {34, 97, 53, 88, 21}.

Name:	findIndexOfBiggest
Input:	nothing
Output:	int indexOfBiggest
Action:	finds and returns the index of the largest element in theArray. So, if the array stores {34, 88, 53, 97, 21} then 3 should be returned.

Name:	findIndexOfSmallest
Input:	nothing
Output:	int indexOfSmallest
Action:	finds and returns the index of the smallest element in theArray. So, if the array stores {34, 88, 53, 97, 21} then 4 should be returned.

Name:	appendArray
Input:	int[] otherArray
Output:	nothing
Action:	increases the storage capacity of the array by the size of the otherArray passed in. This should be done by creating a new array that has a total number of elements equal to its old size, copying over all the old values at the beginning, and then copying over all of the elements from the otherArray at the end. Then assigning the new array to theArray. So, if the old array stored {1, 2, 3} and the array passed in is {4, 5} then theArray will store {1, 2, 3, 4, 5} afterwards.

Level 2

Complete the following methods in the ArrayContainer:

Name:	reverse
Input:	nothing
Output:	nothing
Action:	reverses all the elements in the array. So, if the array stores {34, 88, 53, 97, 21} originally then the array should store {21, 97, 53, 88, 34} afterwards.

Name:	findBiggestValue
Input:	nothing
Output:	int biggest
Action:	finds and returns the largest element in theArray. So, if the array stores {34, 88, 53, 97, 21} then 97 should be returned.

Name:	findSmallestValue
Input:	nothing
Output:	int smallest
Action:	finds and returns the smallest element in theArray. So, if the array stores {34, 88, 53, 97, 21} then 21 should be returned.

Name:	findRangeOfValues
Input:	nothing
Output:	int smallest
Action:	finds and returns the range of values stored in theArray. So, if the array stores {34, 88, 53, 97, 21} then 76 should be returned (97-21).

Name:	appendArrayAtFront
Input:	int[] otherArray
Output:	nothing
Action:	increases the storage capacity of the array by the size of the otherArray passed in. This should be done by creating a new array that has a total number of elements equal to its old size, copying over all the old values at the end, and then copying over all of the elements from the otherArray at the beggining. Then assigning the new array to theArray. So, if the old array stored {1, 2, 3} and the array passed in is {4, 5} then theArray will store {4, 5, 1, 2, 3} afterwards.

Level 3

Complete the following methods in the ArrayContainer:

Name:	shiftLeft
Input:	nothing
Output:	nothing
Action:	shifts all the elements in the array to the left note that the last element on the left loops back to the far right side. So, if the array stores {34, 88, 53, 97, 21, 76} then the array should store {88, 53, 97, 21, 76, 34} afterwards.

Name:	findIndexOfBiggestInRangeOfIndexes
Input:	int startIndex, int endIndex
Output:	int indexOfBiggest
Action:	finds and returns the index of the largest element in theArray within the startIndex and endIndex inclusive. So, if the array stores {34, 88, 53, 97, 21}, 0 gets passed in for startIndex, and 2 gets passed in for endIndex then 1 should be returned.

Name:	findIndexOfSmallestInRangeOfIndexes
Input:	int startIndex, int endIndex
Output:	int indexOfSmallest
Action:	finds and returns the index of the smallest element in theArray within the startIndex and endIndex inclusive. So, if the array stores {34, 88, 53, 97, 21}, 0 gets passed in for startIndex, and 2 gets passed in for endIndex then 0 should be returned.

Name:	getSubArray
Input:	int index1, int index2
Output:	nothing
Action:	returns a new array that contains all the elements from theArray from index1 to index2 inclusive. So, if the old array stored {1, 2, 3, 4, 5, 6, 7}, and 3 is passed in for index1, and 5 is passed in for index2, then {4, 5, 6} will be returned.

Level 4

Complete the following methods in the ArrayContainer:

Name:	shiftRight
Input:	nothing
Output:	nothing
Action:	shifts all the elements in the array to the right note that the last element on the right loops back to the far left side. So, if the array stores {34, 88, 53, 97, 21, 76} then the array should store {76, 34, 88, 53, 97, 21} afterwards.

Name:	findBiggestValueInRangeOfIndexes
Input:	int startIndex, int endIndex
Output:	int biggest
Action:	finds and returns the largest element in theArray within the startIndex and endIndex inclusive. So, if the array stores {34, 88, 53, 97, 21}, 0 gets passed in for startIndex, and 2 gets passed in for endIndex then 88 should be returned.

Name:	findSmallestValueInRangeOfIndexes
Input:	int startIndex, int endIndex
Output:	int smallest
Action:	finds and returns the smallest element in theArray within the startIndex and endIndex inclusive. So, if the array stores {34, 88, 53, 97, 21}, 0 gets passed in for startIndex, and 2 gets passed in for endIndex then 34 should be returned.

Name:	findRangeOfValuesInRangeOfIndexes
Input:	int startIndex, int endIndex
Output:	int smallest
Action:	finds and returns the range of values stored in theArray within the startIndex and endIndex inclusive. So, if the array stores {34, 88, 53, 97, 21}, 0 gets passed in for startIndex, and 2 gets passed in for endIndex then 54 should be returned (88-34).

Name:	removeElements
Input:	int n
Output:	nothing
Action:	eliminates n elements from end of the array. So, if the old array stored {1, 2, 3, 4, 5, 6, 7}, and 3 is passed in for n, then theArray will store {1, 2, 3, 4} afterwards.

Name:	removeElementsFromFront
Input:	int n
Output:	nothing
Action:	eliminates n elements from start of the array. So, if the old array stored {1, 2, 3, 4, 5, 6, 7}, and 3 is passed in for n, then theArray will store {4, 5, 6, 7} afterwards.

Level 5

Complete the following methods in the ArrayContainer:

Name:	move
Input:	int index1, int index2
Output:	nothing
Action:	moves an element from one index to another note that this is different from swap the element is moved over but all the other elements shift to make room. So, if the array stores {34, 88, 53, 97, 21, 76} and 1 gets passed in for index1 and 4 gets passed in for index2 then the array should store {34, 53, 97, 21, 88, 76}. But, if the array stores {34, 88, 53, 97, 21, 76} and 4 gets passed in for index1 and 1 gets passed in for index2 then the array should store {34, 21, 88, 53, 97, 76}.

Name:	swapLargestToFront
Input:	nothing
Output:	nothing
Action:	finds the largest element and swaps it with the element at the start of the array. So if theArray stores {34, 88, 53, 97, 21}, then afterwards it will store {97, 88, 53, 34, 21}.

Name:	swapLargestToBack
Input:	nothing
Output:	nothing
Action:	finds the largest element and swaps it with the element at the end of the array. So if theArray stores {34, 88, 53, 97, 21}, then afterwards it will store {34, 88, 53, 21, 97}.

Name:	removeElements
Input:	int index1, int index2
Output:	nothing
Action:	eliminates all the elements from index 1 to index2 from theArray. So, if the old array stored {1, 2, 3, 4, 5, 6, 7}, and 3 is passed in for index1, and 5 is passed in for index2, then theArray will store {1, 2, 3, 7} afterwards.

Level 6

Complete the following methods in the ArrayContainer:

Name:	moveLargestToFront
Input:	nothing
Output:	nothing
Action:	finds the largest element and moves it to the start of the array. So if theArray stores {34, 88, 53, 97, 21}, then afterwards it will store {97, 34, 88, 53, 21}.

Name:	moveLargestToBack
Input:	nothing
Output:	nothing
Action:	finds the largest element and moves it to the end of the array. So if theArray stores {34, 88, 53, 97, 48, 21}, then afterwards it will store {34, 88, 53, 48, 21, 97}.

Name:	removeSubArray
Input:	int index1, int index2
Output:	int[] removedElements
Action:	returns a new array that contains all the elements from theArray from index1 to index2 inclusive and eliminates those elements from theArray. So, if the old array stored {1, 2, 3, 4, 5, 6, 7}, and 3 is passed in for index1, and 5 is passed in for index2, then {4, 5, 6} will be returned and theArray will store {1, 2, 3, 7} afterwards.

Name:	appendArrayAtIndex
Input:	int[] otherArray, int index
Output:	nothing
Action:	increases the storage capacity of the array by the size of the otherArray passed in. This should be done by creating a new array that has a total number of elements equal to its old size, copying over all the old values up to the index passed in, and then copying over all of the elements from the otherArray after that, and then copying over remaining elements from the original array. Then assigning the new array to theArray. So, if the old array stored {1, 2, 3}, and the array passed in is {4, 5}, and the index passed in is 1 then theArray will store {1, 4, 5, 2, 3} afterwards.