Assignment 5: overloading member functions.

Overview:

Your goal is to create an 'Array' class that is able to hold multiple integer values. The 'Array' class will be given functionality through the use of various overloaded operators

You will be given the `main()` function for your program and must add code in order to achieve the desired result. ***Do not change any code in main()***. If something is not working, you must change your own code, not the code in main().

main()

```
int main() {
	Array arr1(5), arr2(10);

	for (int i = 0; i < arr1.getSize(); i++)
		arr1[i] = i;

	for (int i = 0; i < arr2.getSize(); i++)
		arr2[i] = i;

	cout << "arr1 contains: " << arr1;
	cout << "arr2 contains: " << arr2;

	Array arr3(arr1);
	cout << "arr3 contains: " << arr3 << endl;

	arr2 = arr2;
	cout << "arr2 contains: " << arr2;

	arr3 = arr2;
	cout << "arr3 contains: " << arr3 << endl;

	cout << boolalpha; //Display booleans as 'true' or 'false' instead of 1 or 0
	cout << "arr2 == arr3: " << (arr2 == arr3) << endl;
	cout << "arr1 == arr3: " << (arr1 == arr3) << endl;
	cout << "arr1 < arr3: " << (arr1 < arr3) << endl << endl;

	arr3[0] = 100;
	cout << "New arr3: " << arr3;
	cout << "arr2 == arr3: " << (arr2 == arr3) << endl;
	cout << "arr1 == arr3: " << (arr1 == arr3) << endl;
	cout << "arr1 < arr3: " << (arr1 < arr3) << endl << endl;

	arr1 += arr2;
	cout << "arr1 += arr2: " << arr1 << endl;

	cout << "!arr1: " << !arr1;
	cout << "*arr1: " << *arr1 << endl << endl;

	cout << "arr1++: " << arr1++;
	cout << "arr1 is: " << arr1 << endl;

	cout << "--arr1: " << --arr1;
	cout << "arr1 is: " << arr1 << endl;

	cout << "Total number of elements in all arrays: " << Array::getNumberOfElements() << endl << endl;

	return 0;
}
```

Below is your desired output:

```
arr1 contains: 0 1 2 3 4
arr2 contains: 0 1 2 3 4 5 6 7 8 9
arr3 contains: 0 1 2 3 4

arr2 contains: 0 1 2 3 4 5 6 7 8 9
arr3 contains: 0 1 2 3 4 5 6 7 8 9

arr2 == arr3: true
arr1 == arr3: true
arr1 < arr3: false

New arr3: 100 1 2 3 4 5 6 7 8 9
arr2 == arr3: false
arr1 == arr3: false
arr1 < arr3: true

arr1 += arr2: 0 1 2 3 4 0 1 2 3 4 5 6 7 8 9

!arr1: 5 3 9 8 6 3 2 0 4 7 2 0 1 4 1
*arr1: 0

arr1++: 5 3 9 8 6 3 2 0 4 7 2 0 1 4 1
arr1 is: 6 4 10 9 7 4 3 1 5 8 3 1 2 5 2

--arr1: 5 3 9 8 6 3 2 0 4 7 2 0 1 4 1
arr1 is: 5 3 9 8 6 3 2 0 4 7 2 0 1 4 1

Total number of elements in all arrays: 35
```

Specifications

- The 'Array' class should have an overloaded constructor that accepts an integer argument which represents the size of the array. Use this size to create a dynamically-allocated array of integers.
    - This means you will need at least two member variables for your class: a pointer to a dynamically-allocated array of integers and the size of the array

- The equality operator (==) should return true if one Array object is an in-order subset of another Array object.
    - For example, if a1 = [1, 2] and a2 = [1, 2, 3, 4], a1 == a2 should return true because a1 is contained in a2.
    - if a1 = [2, 1] and a2 = [1, 2, 3, 4], a1 == a2 should return false. Even though a2 has all the elements of a1, they are not in the correct order

    - Notice that you are able to compare two Array objects with differently-sized arrays. Your code should not assume that one array will always be larger than the other.

- The less-than operator (<) should compare two Array objects element-by-element, starting from the beginning. To illustrate...
    - If a1 = [1, 2] and a2 = [2, 1], a1 < a2 would return true because the first element in a1 is less than the first element in a2
    - If a1 = [2, 2] and a2 = [1, 1], a1 < a2 would return false because the first element in a1 is greater than the first element in a2
    - If a1 = [1, 2] and a2 = [1, 3], a1 < a2 would return true. The first elements of a1 and a2 are the same, so move onto the next element. The second element of a1 is less than the second element of a2, so true is returned. If all elements in a1 and a2 are the same, return false.

- The not operator (!) should randomly shuffle the elements in the array
    - Randomly shuffling elements in an array is a logic problem that may take some time to work out. Create a sample .cpp file where you can test your ideas. Once you have the algorithm figured out, transfer it into your 'Array' class.
    - If you can't think of a way to shuffle the elements, the Fish-Yates Shuffle is a simple and efficient way to shuffle. See this link for help.
    - Because you need to randomly shuffle the elements, you will need to use rand() in your code. Remember that when using rand(), you need to call srand() **only once** in your code. Because you can't change main, you will need to think about how to implement this

- The indirection operator (*) should return the smallest value inside the Array object

- The compound sum assignment operator (+=) should extend the left-hand object to include the elements of the right-hand object
    - For example, assume a1 = [1, 2] and a2 = [1, 2, 3, 4]. The following statement:

      a1 += a2

      Should result in a1 = [1, 2, 1, 2, 3, 4] and a2 = [1, 2, 3, 4]. Notice that the right-hand object does not change
    - Because your 'Array' class stores the elements in a dynamically-allocated array, you will need to allocate more room so that the array can hold the additional elements

- You will need to keep track of the total number of elements in all the arrays combined. For example, if a1 = [1, 2]; a2 = [1, 2, 3]; and a3 = [4, 3, 2]; the total number of elements would be 8.
    - Make sure you update this value whenever you change/create an array
    - Pay attention to how the `getNumberOfElements()` method is called in main. This should give you a hint as to how you will implement this feature. Remember that you cannot modify main() in any way!

- Note that this list does not contain every operator and item you will need to add to your class; it only lists the ones that are supposed to do “confusing” things. You *will* need to add more to your class in order to get main() to run

- **You should have your code in three separate files; main.cpp, Array.h and Array.cpp. Zip your project and upload the zipped file.**

Tips

- Do not let the size of main() intimidate you. Comment everything out in main and work on one thing at a time. When you finish creating an overloaded operator, uncomment its section in main, then see if it works. Only once it works should you move onto another overloaded operator
- Look at main() and the desired output as a guide for what operators you need to overload and what they need to do
- **Remember not to change main() at all!**