Homework #6 – Array Class Template

In this assignment you are asked to implement a class template representing an array of a given number of elements of some type. The following UML class diagram shows the attributes and behaviors of the `Array` class template.

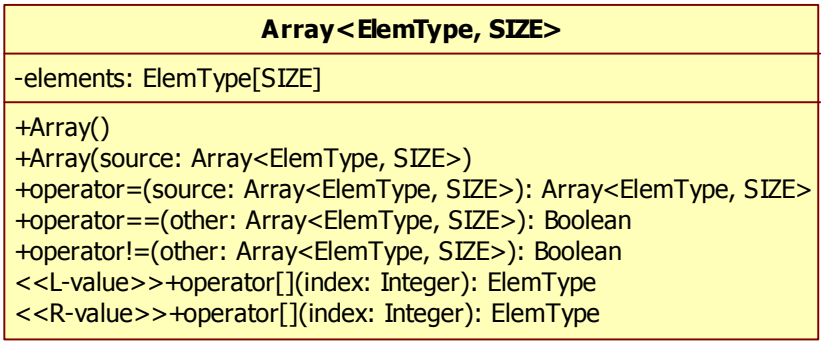

Figure 1. UML class diagram for Array class template

The following code demonstrates how this class template will be used:

```
Array<int, 5> arrayOfFiveInts;  // Define Array containing 5 ints
```

Notice that the type and number of elements are specified when the variable is defined.

1. **(1 point)** Meet these basic requirements:
   a. All non-test code must be implemented in a namespace based on your first and last name (e.g. "`RayMitchell`").
   b. The `Array` class template must be defined (both the class definition and member function definitions) in a file named "`Array.h`". Remember, the entire template is required to be present in the header file because the template source code must be available to the compiler when the template is used by other translation units.
   c. Make sure `const` is used correctly everywhere within the `Array` class template. Be sure to check all pointer parameters, reference parameters, and member functions for proper "const-ness".

2. **(1 point)** Define the `Array` class template to contain a private data member named "`elements`" of type "array of `SIZE` elements of type `ElemType`".

3. **(1 point)** Implement a default constructor for the `Array` class template. This constructor should leave the underlying elements with their default value.

4. **(1 point)** Implement a copy constructor for the `Array` class template. This constructor should copy all of the elements from the `source Array` to the `Array` being constructed.

5. **(1 point)** Implement the copy assignment operator (`operator=`) for the `Array` class template. This constructor should copy all of the elements from the `source Array` to the `Array` on the left side of the assignment. Be sure to properly handle all standard copy assignment situations including preventing self-assignment, and returning the destination `Array` to allow for chained assignments.

6. **(1 point)** Implement the equality (`operator==`) and inequality (`operator!=`) operators for the `Array` class template.

7. **(1 point)** Implement an L-value version of the subscript operator (`operator[]`) for the `Array` class template. Calling this operator should result in a reference to the underlying element being returned allowing the caller to change the element's value within the `Array`. *This operator should throw an `invalid_argument` exception if the subscript index is out of range.*

8. **(1 point)** Implement an R-value version of the subscript operator (`operator[]`) for the `Array` class template. Calling this operator should result in the value of the underlying element being returned allowing the caller to read but not modify the element's value. *This operator should throw an `invalid_argument` exception if the subscript index is out of range.*

9. **(1 point)** Implement a test program in a file named "`hw6.cpp`". Your test program should demonstrate the following:
    a. Create an `Array` using the default constructor
    b. Modify all of the elements of an `Array` using the L-value subscript operator
    c. Output all of the elements of an `Array` using the R-value subscript operator
    d. Create a `const Array` from another `Array` using the copy constructor
    e. Assign an `Array` to an existing `Array` using the copy assignment operator
    f. Compare two `Array`s using the equality operator
    g. Compare two `Array`s using the inequality operator
    h. Demonstrate an `invalid_argument` exception being thrown and caught when the L-value subscript operator is accessed with an `index < 0`
    i. Demonstrate an `invalid_argument` exception being thrown and caught when the L-value subscript operator is accessed with an `index >= SIZE`
    j. Demonstrate an `invalid_argument` exception being thrown and caught when the R-value subscript operator is accessed with an `index < 0` (you can force the R-value subscript operator to be called by accessing an element in the `const Array`).
    k. Demonstrate an `invalid_argument` exception being thrown and caught when the R-value subscript operator is accessed with an `index >= SIZE` (you can force the R-value subscript operator to be called by accessing an element in the `const Array`).

10. **(1 point)** Make sure your source code is well-commented, consistently formatted, uses no magic numbers/values, follows a consistent style, and is ANSI-compliant.

Place all source code and a screen capture of the output produced by your program in a single Word or PDF document. Submit this document.