**Project6**

**Project Six**

Due: Monday 11/30/15 by 11:55PM

**Objective:**

1. Become familiar with the use of [Pointers](https://dropbox.cse.sc.edu/mod/quiz/view.php?id=1359) in C++
2. Develop familiarity of using functions to process [Pointers](https://dropbox.cse.sc.edu/mod/quiz/view.php?id=1359)

**Description:**  
Develop the object *intArray* that creates and manages an array of integers by the use of [pointers](https://dropbox.cse.sc.edu/mod/quiz/view.php?id=1359). The intArray object will implement a number of useful manipulation of arrays including insert, delete, add, equal, and etc.  These functions will limit the use of an array to proper handling. For example these functions will eliminate the possibility of overstepping the boundaries of an array.

**Specific Requirements:**

Your object should satisfy the following criteria:

1. Should be named ***intArray***
2. Should contain a default constructor that will set the data pointer to *null* and size to 0.
3. Should contain the alternate constructor *intArray(int arraySize, int initValue)* that will accept initial size and value for the array. If the initial value is missing, a default value of 0.
4. Should contain the alternate constructor *intArray(int anArray[], int arraySize)* that will accept an array and create an object equal to the array.
5. Should contain a copy constructor.
6. The parameter that holds the size of the array should be private.
7. Should contain a *size()* method that will return the size of the array.
8. Should contain a *set(index, value)* method that will set the data[index] = value.
9. Should contain a *get(index)* method that will return data[index].
10. Both *set* and *get* methods should exit the program if index is outside of the array boundaries.
11. Should contain an overloaded assignment operator (=) similar to your previous *copy(intArray anIntArray)* method that will copy one *anIntArray* into another. Furthermore the operator= should be capable of cascaded operation.
12. Should contain an overloaded equality operator (==) similar to the *equal(intArray anIntArray)* method that will return true if two arrays are equal in size and content.
13. Should contain an overloaded addition and subtraction operator (+, -) similar to your previous *add(intArray anIntArray)* and *subtract(intArray anIntArray)* methods that will add or subtract one *intArry* into/from another. Both operations should be capable of cascaded operation.
14. Should contain overloaded post and pre-increment operators. Should be capable of cascaded operation.
15. Should contain overloaded unary - operator. Should be capable of cascaded operation.
16. Should contain overloaded I/O-insertion ( << ) and I/O-extraction ( >> ) operators. Examples: cout << intArray should result in printing of the array to the stdout. cin >> intArray should prompt the user to entry values for every element of the array. Both operations need to be cascade enabled.
17. The *copy*, *add* and *subtract* methods above should exit the program if size incompatibility is identified.
18. Should contain an *insert(index, value)* method that will insert the *value* at the *index* place of an array. The value should be inserted and not replaced.
19. Should contain a *remove(index)* method that will delete the *index* item of the array. Your method should then return the remove value.
20. Both *insert* and *remove* should exit the program if the *index* is invalid, and should expand/shrink the array appropriately.
21. Your program should have no memory leaks!