COM S 213 – Fall 2002

ASSIGNMENT #6: **Dynamic Array Class**

DATE GIVEN: **10/17/02**

DATE DUE: **10/24/02**

PURPOSE:
To review operator overloads, dynamic allocation of arrays and to utilize destructors and the array index operator overload.

ASSIGNMENT:
In our last lecture I prototyped a very simple integer Array class that could be used to do rudimentary bounds checking while still being able to be used "like" an array in the sense that the array index operator was employed to allow us to use the following syntax:

```
MyArray ma;

ma[0] = 5;

cout << ma[0] << endl;
```

This class was very limited as it hard coded the maximum array size at compile time. Your assignment is to come up with a new version of MyArray (called IntArray) which allows the size of the array to be determined at runtime. It will also be a little more flexible than the original MyArray. The tasks you must complete are:

- IntArray will have a single constructor which takes an argument to specify the initial size of the array (to be allocated dynamically)
- IntArray will have a destructor which deallocates memory (if there actually is allocated memory)
- IntArray will have a member function called "resize" which allows the dynamic array to be reallocated on the fly (preserving the contents of the array in doing so)
- Operator[] will be overloaded to allow assignments to array elements and the retrieval of array elements using the syntax: array[i].
- All array element accesses will be checked for an out of bounds condition. If one is detected, a message will be sent to cout indicating the condition.

You will test your code by writing a test program that demonstrates all error checking and normal operations.

SUGGESTIONS

This should be fairly easy, preserving the contents of the array on a resize may be the only potential tricky thing you'll need to do. If you are "downsizing" the array via the resize() method, it's OK to loose the ability to reference the set of items that fall outside the bounds of the newer (smaller) array.