Pass By Reference

This exercise will give you a chance to work with pass-by-reference, one of the first, useful things we can do with pointers. On the course homepage and under the following path, you will find a partial implementation called passByReference.c.

/afs/eos.ncsu.edu/courses/csc/csc230/common/www/sturgill/exercise08

The program won't compile until you add a few missing functions. Once it's working, it should print the following output when run:

```
a = 100 b = 50 c = 25
a = 110 b = 60 c = 35
a = 60 b = 35 c = 110
a = 60 b = 35 c = 109
```

From the source code, you'll see you have to add three functions. You can't modify the contents of main() at all, but by taking parameters passed by address (and maybe some passed by value also), your three functions will be able to change the contents of variables declared in main.

Here's what your functions need to do:

- incrementAll( ) : This function should increment a, b and c by the value of the 4th parameter. The main( ) function uses this to add 10 to each of the variables.
- rotate( ) : This function will simultaneously copy the value of b to a, c to b and a to c. You'll probably want to use a temporary, local variable in the rotate() function to help with this.
- getLargest( ) : This function will take the addresses of a, b and c and it will return the address of the one containing the largest value. This is the first time we've used a pointer type as the return value of a function. You just need to give int * as the return type, and be sure to return one of the addresses passed in (i.e., not the address of a local variable declared in getLargest(); that would be bad).

When you're done submit your completed passByReference.c file to the exercise_08 locker in WolfWare Classic.