CSCI-1200 Computer Science II — Fall 2006
Lab 7 — Complex Number Operators

Checkpoint 1

Please download the code for the `Complex` class we discussed in lecture on Friday:

`http://www.cs.rpi.edu/academics/courses/fall06/cs2/labs/07_operators/complexMain.cpp`
`http://www.cs.rpi.edu/academics/courses/fall06/cs2/labs/07_operators/complex.h`
`http://www.cs.rpi.edu/academics/courses/fall06/cs2/labs/07_operators/complex.cpp`

and then turn off your internet connection. Re-familiarize yourself with the code, then compile and run it. Uncomment the remaining code in the `main()` function and implement and test the missing operators (we did some of this work during lecture).

To complete this checkpoint: Show a TA your implementation.

Checkpoint 2

Implement the following operators for the `Complex` class (or explain why they cannot or should not be implemented). Think about whether they should be non-member, member, or friend.

`negation` `operator==` `operator!=` `operator<`

To complete this checkpoint: Discuss your implementation with a TA.

Checkpoint 3

Let's talk about *templates* (you may remember the term from lecture). STL vectors, lists, and maps are *templatized classes* in that they are able to store things of any type (as long as some basic operations are defined for that type). First we'll look at simple *templatized functions*. The templatized function below takes three arguments, a boolean and two arguments of the type T (which will be specified at runtime). If the first argument is true, the function returns the second argument, otherwise it returns the third argument. Pay special attention to the new syntax.

```
template <class T>
T select(bool flag, T a, T b) {
  if (flag)
    return a;
  else
    return b;
}
```

Type this into your programming environment and try these examples:

```
cout << select(true, 5, 7) << endl;
cout << select(false, "Sally", "Fred") << endl;
```

Now write a new templatized function that takes in an integer n and a vector of the templatized type and returns the element in the vector at index n. Test your program with a variety of inputs (including complex numbers!).

To complete this checkpoint: Show a TA your tested and debugged function.