**CS 2315: Exam 1 Review**

* The member access(.), ternary/conditional(?), scope resolution(::) and pointer(\*) operators cannot be overloaded
* The access-specifier label “public:” contains the keyword public is an access specifier.
* Variables or functions declared after access specifier private (and before the next access specifier) are accessible only to member functions of the class for which they’re declared
* The default access for class members is private so all members between the class header and the first access specifier are private.
* The client of an object is any class or function that calls the object’s member functions from outside the object
* Set functions are also sometimes called *mutators*, and get functions are also sometimes called *accessors*.
* Constructors cannot return values, so they cannot specify a return type (not even void)
* Any constructor that takes no arguments is called a default constructor
* Each of the previous examples in the chapter consists of a single .cpp file, also known as a source-code file, that contains a GradeBook class definition and a main function
* When building an object-oriented C++ program, it’s customary to define reusable source code (such as a class) in a file that by convention has a .h filename extension—known as a header
* Each member-function name is preceded by the *classname::*, which is known as the scope resolution operator
* The source-code must both be compiled, then linked together—that is, the member-function calls in the client code need to be tied to the implementations of the class’s member functions—a job performed by the linker
* data hiding: restricting access to certain members of an object
* const appearing after the parentheses in a member function declaration specifies that the function will not change any data in the calling object
* Compiler error occurs on attempt to access private member using dot operator
* Place class declaration in a header file that serves as the class specification file
* Destructor name is *~classname* (no return type or arguments; one destructor per class)
* If constructor allocates dynamic memory, destructor should release it
* When the object is destroyed, its destructor executes “delete r;”
* The precedence, heirity, and associativity of an operator can’t change
  + Presidence: working from left to right
  + Associativity: if an operator is overloaded for class a, but not for some data type b, then a(operator)b will work, but b(operator)a will not
* Static: constructor only called once
* The operators =,., comma, <<, and >>, as well as some member functions and other stuff I don’t care to look up can be cascaded
* The default constructor/destructor and the equals(=), reference(&), and comma(,) operators are automatically defined for a class
* A friend function is not a member of a class, but has access to the private members of the class it is defined in.
  + A friend function can be a stand-alone function or a member function of another class
* The copy constructor is initialized with a class
* Copy constructors initialize an object containing the data from another object
* Friend functions don’t require *classname::*
* When overloading the iterative operators, you must consider the preincrement and the postincrement
  + classname::operator++() is the preincrement
  + classname::operator++() is the postincrement
* end of stuff

++x != x++

int x=5

++x == 6 -> true

x++ == 6 -> false

constructs from inside-out

destructs from outside-in

members -> statics -> globals

istream & ostream - friends

operators - members or friends

mutators & accessors – members or friends

associativity of an operator can't change

r+s-t

same presidence, so works left to right