Final – Review Session

1. Overloaded operators. Overloaded operators; how to overload each type of the operator: prefix/postfix increment, subscript, assignment, and any other operator. Implement an overloaded operator.
2. Types of polymorphism and what each of them is useful for; make a decision on what to use in a particular case.
3. Virtual functions; their purpose, when they are used; virtual tables; understand what code using (or not using) virtual functions does; implement something using (or not using) virtual functions; virtual destructors, when they are used; understand the code using them; write the code that would (or would not) use them appropriately.
4. Templates. Template functions and classes; know how to define them; what the compiler does with template code; defining member functions for template classes; inheriting from template classes; template specializations, etc.
5. Copy constructors, assignment operators and destructors. What they do, what signature they must have. Know when a copy constructor is called; when an assignment operator is called; when a destructor is called – should either know this or know how to write code to test it. Implement a copy constructor, an assignment operator, and/or a destructor for a class.
6. `this` pointer; what it is, how it is implemented; when and why it is used.
7. Exceptions. What they are used for; what happens when they are thrown. Stack unwinding: how many times it occurs in a given context until the exception is caught. Resource leaking. Exceptions and local variables/pointers. Exceptions in constructors/destructors.
8. Smart pointers; different smart pointer implementations; referencte counting;
9. STL containers. Vector member functions and what they do. Best class to use for a particular problem. Hash functions. What they are used for and why; their desired properties. Identify problems with hash functions and how to fix them, if possible.
10. Assorted best practices. Correct uses for built-in data types, conversions and casts, problems with passing by value, proxy classes, etc.
11. Testing your code. Design a test function or functions for a particular implementation