C++ Midterm Exam Stuff

There will be a lot of short answer on this exam, and a bunch of code (both writing and reading). There will be no trick questions, and I'll stay away from asking about obscure details. The focus will be on the sorts of things you'd need to know in real situations: the kind of code you'd actually see (and write). I've tried to emphasize not just how to write code, but how to write *good* code - why certain practices are good ideas, why others are not good ideas, the kinds of trouble certain features can cause for you - and this may be reflected on the exam.

Below is a large list of questions we've covered so far in the semester. You should (hopefully) know a lot of this already. Not *all* of this will be on the exam, of course! But if you can answer all this stuff, and you've got a good idea of how the C++ features work together, you'll be in great shape.

- What makes C++ a strongly typed language? What are the basic types?
- What are the two kinds of errors you can get when attempting to make a C++ executable?
- What are the rules for naming variables and (most) functions in C++?
- How do you declare variables? What is the difference between a variable's declaration and it's definition?
- How do you declare functions? What is the difference between a functions' declaration and it's definition? How do you call a function?
- What is the difference between the pre-increment and post-increment operators?
- What is the syntax for the various conditional constructs in C++ - if/else, while, for, do-while? What are the differences between these?
- What are the two C++ keywords that can control the execution of a loop from inside that loop, and how do you use them?
- What are the different boolean logic operators used for C++ conditional constructs?
- Why is it important (for you, not the compiler!) to be careful with indentation/spacing?
- What is explicit type conversion? Implicit type conversion? How can you force an explicit type conversion?
- In a C++ program, what order should stuff come to keep the compiler happy?
- What's a header file? What goes in a header? How do you use one and what is it used for?
- What's the difference between pass-by-value and pass-by-reference? What are the pros and cons of each?
- What are default arguments and how do you use them?
- What are overloaded functions? What should you be careful of when using them?
- What is a pointer and how do you declare and use one? What is the term for getting at what a pointer is pointing to? What are the various operators involved in using pointers, and what do they do? What are pointers good for? What are the potential dangers of using them?
- What are references and how do you use them? Under what circumstances would you use a reference instead of a pointer? A pointer instead of a reference? What are the rules for defining and using references? What are the potential dangers of using them?
- How do you declare and use an array? How do you pass one to a function? What's the relationship between arrays and pointers?

- What is pointer arithmetic and how does it work?
- What is scope? What is lifetime? When are these two the same? Different?
- How do you statically declare variables? Dynamically declare them? What are the pros and cons of each of these?
- How do you clean up after dynamically declared variables? What is actually going on when you do this? What are the dangers here?
- What are C-style strings? How do you work with them? What shortcuts exist in the C++ language to help declare and use them? What standard mapping does C++ use to convert between characters and integers? What's an escape sequence and what are they used for?
- What are structures? Classes? The rules for declaring them? The difference between them? How do you access functions and variables contained in a struct or a class?
- What are the different access specifiers we've learned so far? What do they do? Why are they useful? How do you define and use them?
- What (basically) is OOP? What is an abstract data type?
- How do you declare and define a member function, either inside or outside of a class? How does the concept of scope change when we're talking about classes?
- How do you create a new instance of an object? Statically? Dynamically? What is this called?
- What is encapsulation? What's the point?
- How do accessor functions help support the concept of encapsulation? What do we mean when we refer to the interface of a class?
- What is the friend keyword good for? How do you declare a class to be a friend? A function? Does friend help or hurt the encapsulation concept?
- We've talked about three different uses of const - const variables, const methods, and const function parameters. How do you use each of these, and what's the point? Is it worth all the hassle?
- What is a constructor and what does it do? How do you make a constructor? How do you have multiple constructors? When is a constructor called?
- What is a destructor and what does it do? How do you make a destructor? When is a destructor called?
- What is a copy constructor and what does it do? How do you make one? What's the point of a copy constructor?
- What is operator overloading and how do you do it? Why are they useful? Why is operator overloading sometimes not a good idea?
- What sorts of functions does C++ automatically provide for a class, behind the scenes, if you don't explicitly provide them yourself? Why is it sometimes necessary to write these functions yourself?
- What is a type conversion operator and how do you make one? Why are they useful?
- How do you separate your source code into multiple files? Why do we care?
- What does the keyword "extern" do when applied to a variable? Why is that useful?
- What is the "this" keyword? What type is it? What is it used for?
- What is a static variable? A static member function? Can a static member function access regular data? Can a regular member function access static data? How do we access static data/functions even when no classes of that type have been instantiated?