Midterm Review

Basics

The midterm will be Friday, March 7th at 11:00 am – normal class time. What is not normal is that it will be at the Nursing Computer Lab (6th floor, nursing building).

You will have 40 minutes for the exam, just to give you a little bit of leeway when it comes to hiking across campus. The exam is designed with this time limitation in mind. The time is strictly monitored, though – the system will lock you out 40 minutes after *you* start your exam. This is done to ensure fairness; I can't keep the lab past noon, and some people will inevitably be late as a result of distance from the rest of campus. This way everyone gets the same amount of time.

What you need to know

For starters, it's a good idea to read (or, really, skim) everything that *I* have written – the practicum assignments, homeworks, etc.

Notably:

- PA1/PA1.2
  - Understand how the recursion works for parsing the XML
  - Understand how the project takes advantage of both inheritance of interface and inheritance of behavior
- PA2
  - Understand what the testing portion of the assignment is and why you're doing it

Specific topics

- Everything is an expression
  - Expressions have types
  - Expressions have values
- map
- How engineering is a separate topic from programming
- Core ideas of object oriented development
  - Encapsulation
  - Aggregation/"Has a" relationship
  - Inheritance/"Is a" relationship
    - Inheritance of interface

        - Inheritance of behavior[1]
- Dijkstra's argument against the goto statement
- Include guards
    - How they work
    - *Why* they work
- Calling code from a base class
- XML
    - Mostly basics
    - Be able to identify bad XML markup
- Function pointers
    - Concept of being able to store logic in a variable
    - Method evaluation operator: ()
- Code reuse
    - e.g., what separates "good code reuse" from "copy, paste, and edit a bit"
- Makefiles
    - Understand how it works recursively with regard to dependencies
    - Understand how linking works in a makefile
- Build process/toolchain
    - Preprocessor
    - Compiler
    - Assembler
    - Linker
- Software engineering models
    - Waterfall
    - Agile
    - Rough timeframe for development of the models
- Virtual and non-virtual methods
- Parameter passing semantics
    - Pass by value
    - Pass by reference
- Singletons
- Static methods/member variables/variables
- Private constructors
- Testing
    - System testing
    - Unit testing
    - Regression testing

---

1 This is often seen as "inheritance of implementation", but any use on the midterm will follow "inheritance of behavior" as I've used in class. They mean the same thing, of course.

What you don’t need to know

Any history covered in class, with the exceptions mentioned above.

You will not be asked to write any code[2], but you will need to be able to read code. You *will* need to know a lot of stuff from CS215 to be able to read the code presented -- <iostream> objects, vectors, etc. Make sure you can make sense of the code examples posted on the web site.

2 I don't consider it reasonable to ask someone to write correct code – at least, at the level of complexity necessary to ask a reasonable question at this level – without having a compiler present.