CMSC 202 Midterm 1 Topics Fall 2015

C syntax

- basic data types; declaration and initialization of variables
- type casting (static_cast and "old style")
- constants (declaration using const; global constants)
- arrays (declaration and initialization; usage; passing to functions)
- operators (binary, relational, unary; shorthand operators)
- blocks and scope
- if-else if-else
- switch
- do, do-while, for (syntax and appropriate use)
- break and continue
- user-defined functions
    - prototype vs. implementation (syntax, placement)
    - parameters vs. arguments
    - return and return values
- C-strings (declaration and initialization; as char[]; importance of null terminator)
- Basic C-string functions (strnlen, strncmp, strncat)
- Pointers
    - relationship between pointers and arrays; arrays in memory
    - pointer variables (declaration and usage)
    - use of '*' and '&' operators
    - pointer arithmetic
    - new and delete (purpose; syntax for basic types and arrays)
- Pitfalls
    - "=" vs. "==", especially in conditionals
    - forgetting to use { } with conditional or loop
    - loop and array bounds, e.g.,
        - for (i = 0; i <= NUMDATA; i++), should be "<"
        - array indices from 0 to length of array - 1, not 0 to length of array

C/C++ Program structure

- basic program layout (single file)
    - #include (usage, placement)
    - using namespace std;
    - location of prototypes, main, function implementations
- program layout (multiple file)
    - what goes in the header (.h) file?
    - guarding the header file (how and why)
    - what goes in the implementation (.cpp) file?
    - use of #include in include header file

C++ Syntax and Libraries

- iostream library (usage of <<, >>, cin, cout, cerr, endl)
- string class (basic usage of string type)

Classes and Objects

- what is a class? what is an object? contrast with struct
- encapsulation
- class interface (syntax; read and understand simple interface)
- class implementation syntax (syntax; read and understand simple implementation)
- use of dot (.) to access object's functions or variables
- private and public (purpose and syntax; standard usage; access rules)
- accessors, mutators, facilitators (purpose; read and identify)

CMSC 202 Coding Standards

- variable, constant, function, and class names
- function header comments (pre- and post-conditions)
- appropriate use of in-line comments