COMP 2404A Midterm Review – Winter 2020

- Section 1 -- Basics of C++ development
    - Linux platform
        - type of shells
        - basic shell commands
        - program building:
            - compiling, linking, differences between the two
            - compiling translates source code into object code
            - linking translates and combines object code into an executable
            - Makefiles
    - Basic language features
        - variables
        - functions
            - global function, member functions
            - function declaration vs implementation
            - function design
        - types of parameters
            - input, output, input-output (NOT ANYTHING TO DO WITH I/O)
            - only parameters have this kind of "job"
        - parameter passing
            - pass by value, pass by reference by reference, pass by reference by pointer
        - operators: operands, arity (# of operands), precedence, associativity
        - expressions, statements, blocks, scope (local vs global)
        - references: what are they, what are they not
    - Programming conventions
        - naming conventions: constants, variables, functions, data types
        - indentation, commenting
    - Class definition
        - super star operator: binary scope resolution operator
        - access specifiers (public, protected, private)
        - code organization: what goes in a header file, what goes in a source file
        - class interface: set of public members of a class
        - include guards
        - variable scope: block scope, file scope
        - Namespaces

- Constructor and destructors
  - default arguments
  - default constructor: a constructor that has no parameters; when is it called
  - destructors: what are they, when are they called
  - copy constructors: what are they, when are they called
  - conversion constructors: what are they, when are they called

- Memory management
  - stack vs heap
  - function call stack, stack frames
  - pointers
    - what are they, why are they used, how are they used
    - main operators: address-of (&), dereferencing (*), arrow
    - differences with references
  - parameter passing with pointers
  - memory allocation: statics vs dynamic
  - memory leaks: what are they, how do they happen
  - dynamic memory allocation: new, delete
  - **4 different kinds of arrays**
    - dynamically allocated vs statically allocated arrays
    - arrays of objects vs arrays of object pointers
    - how to allocate and deallocate all 4 kinds of arrays
    - example: Date** xyz; xyz = new Date*[MAX];
      xyz[0] = new Date; xyz[0]->print(); delete xyz[0]; delete [] xyz;

- Section 2 -- Basics of object-oriented design

  - OO design overview
    - software engineering life cycle activities
    - OO design principles
    - data abstraction: making class interfaces simple, separating interfaces from implementation
    - encapsulation: grouping together data and behaviour that belongs together
    - principle of least privilege

  - Object design categories:
    - types of object categories: entity objects, control objects, boundary objects (view objects), collection objects
    - what are they, why do we separate the different kinds

- **UML class diagrams**
  - classes: attributes, operations, parameters, parameter type (in, out, inout), access specifiers (#, -, +)
  - associations (relationships): inheritance, composition
  - composition: multiplicity, directionality
  - do not show: getters, setters, constructors, destructors, collection objects, and do not show objects as attributes (use associations instead)

- Section 3 -- Essential object-oriented techniques

  - Encapsulation
    - composition: member initializer syntax, constructor and destructor order of execution
    - constants (objects, data members, member functions)
    - friendship
    - static class members
    - **linked lists**: singly linked, doubly linked, with or without tail, addFront(), addBack(), addInOrder(), addInPosition(), removeFront(), removeBack(), removeFromPosition(), removeElement(), traversing, cleanup

Midterm:

- covers everything up to and including section 3.1 (encapsulation)
- 80 minutes
- out of 50 marks
- concept exercises (UML, 2 questions): 20 marks
  - given UML, write classes – actual class definitions, no pseudo-code
  - given classes, draw the UML
- programming (3 questions): 30 marks
  - no pseudo-code, you must write actual C++ code

BRING:

- campus card
- **pencils**, erasers, ruler
- go to the bathroom BEFORE the midterm