T-501-FMAL Programming languages, Topics to Revise for the Final Exam Spring 2021

The exam will be a home exam in the form of Canvas assignment(s). It will consist of multiple-choice and free-text questions about concepts and questions on code (understand what a piece of code does, write some code).

Please revisit the following topics. Please read the book.

- F# (Appendix):
  - values in numerical types, boolean type, unit, tuple and record types, operations on them
  - first-order vs higher-order functions
  - multiple-argument functions in uncurried, curried form
  - recursion
  - parametric polymorphism
  - option type, list type, user-defined discriminated union types (for trees etc)
  - map, filter, fold for lists
  - exceptions
  - references
- Expressions (Ch 2-3):
  - simple expressions with global variables only
  - concrete syntax, abstract syntax
  - lexing
  - parsing
  - interpretation of simple expressions
  - environments
  - a stack machine for simple expressions
  - compilation of simple expressions
  - compilation correctness
  - expressions with local variables (let-blocks)
  - interpretation and compilation of expressions with let-blocks
- Functional programming (Ch 4-6):
  - a first-order functional language (FirstFun): expressions with function variables and function calls
  - static vs dynamic scope rule
  - closures as function values
  - interpretation of FirstFun
  - a higher-order functional language (HigherFun): expressions with first-class functions
  - anonymous functions
  - interpretation of HigherFun
  - type schemes, polymorphic type inference
  - unification of types
  - untyped lambda-calculus, terms, alpha-conversion
  - capture-avoiding substitution, beta-reduction
  - the Y combinator

  - reduction strategies
  - simply-typed lambda calculus, types, type derivations
  - polymorphically-typed lambda calculus
  - encodings of booleans and natural numbers

- Imperative programming (Ch 7-10):
  - a naive imperative language (Imp): mutable variables, expressions and statements (aka commands)
  - naive stores
  - interpretation of Imp
  - a C-like imperative language (MicroC): pointer and array variables, access expressions
  - two-level stores (environment + store)
  - l-values (locations), r-values (contents of locations)
  - call-by-value and call-by-reference parameter passing
  - interpretation of MicroC
  - frames (activation records)
  - a stack machine for MicroC
  - compilation of MicroC
  - real-life virtual machines JVM and CLR
  - heap, programmed deallocation
  - freelist
  - garbage collection (automatic deallocation): reference counting, mark-and-sweep, two-space stop-and-copy
  - a C-like imperative language with heap (ListC)