Grading and Course Learning Outcomes

A large number of fundamental skills and concepts are introduced in this course. Grades reflect students' ability to apply them towards competently implementing solutions. When skills are first introduced, they will generally be measured in isolation. We later measure them through their contribution to students' construction of correct solutions.

Student mastery of learning outcomes are measured using the following scale:

- Insufficient: no credit
  - 0: no evidence of familiarity
  - 1: familiar, but not proficient
- Sufficient: full credit
  - 2: proficient: you can reliably apply this learning outcome towards solving or analyzing a problem
  - 3: exceptional

A learning outcome is satisfied when a student can effectively apply it towards the solution of realistic problems. This includes:

- Selecting appropriate problem-solving techniques
- Applying them proficiently in developing solutions that are correct and comprehensible by others

An elementary example: if a problem involves computing a sum of numbers in various radixes, full credit will be assigned for a solution where

- values are converted to a common radix
- addition is utilized and applied appropriately
- an expert can easily discern how the student solved the problem and that the solution is correct

In important note: A student may find a particularly insightful way of simplifying a problem. If the grader observes this, they will assign the exceptional score of "3." While such scores reflect greater maturity than is required for this course, no "extra credit" is assigned.

Learning outcomes are divided into families. Students' grades are assigned based upon the the fraction of "families" of learning outcomes for which students reliably demonstrate sufficient mastery to effectively utilize them in constructing solutions to realistic problems.

To assist students in identifying particular strengths and weaknesses, mastery of recently introduced skills will be identified individually rather than by family.

family	Prerequisite knowledge from CS1,CS2, digital design, discrete, and precalc	Students will be familiar with...	Students will be able to effectively apply skills...	Students will be able to analyze & synthesize solutions..
NR: numeric representation & ops	- familiar with radixes, signed representations,		- convert hex, dec, signed-dec, binary binary metric - signed/unsigned	- can determine appropriate representations for elementary types and design low-level programs

	and scientific notation		comparison (flags,order) - Add-with-carry - cast/sign-extend - floating point	that compute arithmetic results
L: linearization	- algebra, - arithmetic & control-flow structures of an oo language - block structures,		- expressions (incl side effects) - control flow (if/while/for) - translate boolean logic - branch tables - op on arrays, structs, and pointers	- can translate infix expressions and block-structured programming constructs to assembly language
GA: gross architecture	- able to program in at least one oo language - familiar with cobinational and sequential logic	Can describe the fetch-execute cycle in the context of the roles of PC, SP, flags, registers and memory.	- select appropriate instructions - specify operand order - encode and decode instructions - specify addressing mode - utilize interrupt mechanism - implement interrupt handlers	- can implement and debug simple imperative programs in assembly or machine language
Ti: timing	- algebra - synchronous logic - frequency		- determine cycles/instruction - determine which instructions repeat in a loop	- can compute the execution time of a simple loop - can design a loop that delays a specified amount of time
Sub: subroutine linkage & separate compilation	- in oo languages studied in CS1/2		- parameter passing - return value - allocation of auto vars - register usage - global/local symbols	- can write or call a method with local variables, parameters, and return value in assy lang
VA: variable allocation	- in oo languages studied in CS1/2		Can define and use variables with various.. - scope: visibility (file/method/program) - variable lifetime (program/method) - size - alignment - arrays - pointers - structs	Can appropriately allocate static and auto variables including arrays and pointers in assembly language
T:	- ide		Can effectively employ in	Can compose and debug

tools	- hierarchical filesystems		the composition and debugging of programs - editor - compiler - make - svn - bash - gdb - source code repo	simple programs in a command-line environment
WC: written communica-tion	- proficient in english		Can - interpret technical documentation on familiar topics - describe implementations that they design - recognize/use technical terminology - appropriate documentation for code	Can appropriately document simple programs
MP: mature programm- ing	- proficient in OO programming - appropriate comments - can modularize - appropriate symbol names - coding style		Can utilize in a program - appropriate comments - modularize - imperative programming - appropriate symbol names - coding styles	Can appropriately modularize and document simple programs consisting of multiple files
Perf: advanced topics for performance		- pipelining - vectoriza-tion - predicated instructions		Can identify when these topics are relevant to constructing an efficient solution.
DEV: devices	gates, latches, (de)multiplexers, ALUs, switches, counters	- gross characteriscs of memory & storage devices - counter-timer	can implement - simple programmed i/o - interrupt handlers	Can design programs that implement simple programmed i/o and interrupt handling - can determine the types of storage devices suitable for a variety of uses.

Two generic families of skills are also measured.

- Clerical accuracy: clerical errors do not significantly interfere with students' ability to construct correct solutions.
- Completion: students are able to construct complete solutions in the allotted time.