Overview

This document is an attempt to build a shared understanding of the knowledge and skills expected from a student who completes the core CS sequence enshrined in the CS MTM.

CS205 was developed intentionally for the MTM and has an existing proscribed list of outcomes (available on the OCCC website). Thus it is not included in this document and topics which it covers (low level & systems programming) are not included here.

Document Tabs/Pages

Programming	These are generally taught in CS161/162, though some topics may drift into or be formalized in data structures
Analysis Collections Algorithms	These are generally considered CS260 topics. However, due to the amount of material present, it is common (if not essential) to begin teaching some of them in CS162.
PlatformsTools	Essential knowledge and skills that are not associated with any particular course but expected from someone who has completed the MTM.

Importance Levels

How important the topic is in the overal curriculum.

Essential	Students should be getting all of these topics. Skipping this is expected to directly affect performance in later courses.
Recommended	Important concepts. However, may not be possible in all languages; may be less critical for future classes.
Elective	Useful, but skippable. May not be possible in all languages. Or may not be foundational for later work.

Proficiency Levels

What level of proficiency is expected at this point in a student's education.

Proficient	Comfortable and confident with these topics. While they may continue to develop this skill/knowledge, they should be entirely comfortable with the topic and will likely not recieve further formal instruction in it.
Competent	Can use these ideas/skills at a basic level, lacks fluency in them. These are often skills expected to be developed futher in later courses.
Exposure	Is aware of general concept. Likely does not have a deep level of understanding or any practical proficiency.

Big O Expectations

A value in this column indicates how fluent students are expected to be reasoning about Big O for this topic, which may differ from their fluency implementing/using the topic.

Proficient	Can make a clear case for why a BigO is what it is and/or analyze new algorithms based on this. Does not imply ability to write a mathematical proof.
Competent	Can use these ideas/skills at a basic level, lacks fluency in them. These are often skills expected to be developed futher in later courses.
Exposure	Is aware of general concept. Likely does not have a deep level of understanding or any practical proficiency.

Command Line Expectations

A value in this column indicates how fluent students are expected to be doing these tasks from the command prompt as opposed to using a GUI interface.

Competent	Can perform these skills from the command prompt. Can do the basics without reference. Likely needs to refer to documentation to do anything complex.
Exposure	Knows it can be done. Would likely need to refer to documentation to do even the basics.

Contributors

Yong Bakos	OSU-Cascades
Lucas Cordova	WOU
Karla Fant	PSU
Breeann Flesch	WOU
Kathleen Freeman	UofO
Tim Harrison	EOU
Gayathridevi Iyer	PCC
Doug Jones	PCC
Troy Lanning	Klamath CC
Andrew Scholer	Chemeketa CC
Robert Surton	Chemeketa CC
Ken Swartwout	COCC
Michal Young	UofO

Category	Outcome	Importance	Proficiency	Dissent
Write and execute a program	Given specifications, write a reasonable-sized program without using external references.	Essential	Proficient	
Write and execute a program	Decompose a program into modules at a source file level. Organize source files using any relevant language features (e.g. including headers or importing modules).	Essential	Competent	
Discuss computing problems among humans	Read an existing program, and explain what purpose it serves, what it does, and how it operates.	Essential	Proficient	
Discuss computing problems among humans	Value consistent programming style. Adapt and to and use existing style of a codebase or team.	Essential	Proficient	
Discuss computing problems among humans	Explain requirements, design decisions, and coding decisions, using source code (variable names, conventions, etc.), programming comments, and documentation.	Essential	Competent	
Discuss computing problems among humans	Justify the way a program is decomposed, by analyzing an existing set of modules to describe its benefits and drawbacks, by arguing for why a decomposition was designed the way it was, and by improving a decomposition when warranted.	Essential	Exposure	
Discuss computing problems among humans	Collaborate with others through code reviews or pair programming.	Essential	Proficient	Not covered at EOU.
Discuss computing problems among humans	Use resources and peers in a way that avoids plagiarism.	Essential	Proficient	
Reason about values in a program.	Distinguish a value, a location that stores a value, and an identifier that names a location or value.	Essential	Proficient	
Reason about values in a program.	Analyze an identifier for type, that is, the set of values it might hold and properties all those values have in common. Use provided features for annotating types.	Essential	Proficient	
Reason about values in a program.	Use references to data, such as with a pointer whose value represents the location of the data, or a reference that is an alias for another identifier for the data. Reason about aliasing.	Essential	Proficient	
Reason about values in a program.	Determine whether an identifier in a program is a variable, which might hold more than one value over the running of the program, or a constant, which always has a fixed value. Use provided features for distinguishing variables and constants.	Essential	Proficient	
Reason about values in a program.	Assign appropriate scope and lifetime to each variable in a program. Use provided features for annotating scope and lifetime.	Essential	Proficient	
Debug a program.	Interact with a program in order to confirm expected behaviors and attempt to trigger faults.	Essential	Proficient	
Debug a program.	Simulate a program's steps mentally or in writing to predict behavior.	Essential	Proficient	
Debug a program.	Add assertions and logging to a program to observe computations and isolate unexpected behavior.	Essential	Proficient	
Debug a program.	Run a program under a debugger to observe computations and isolate unexpected behavior.	Essential	Proficient	
Debug a program.	Recognize common runtime errors and recall how to approach resolving them.	Essential	Proficient	
Debug a program.	Recognize common static errors, including syntax errors, and recall how to approach resolving them.	Essential	Proficient	
Debug a program.	Distinguish logic errors, such as a programmer's mistake, from exceptional conditions, such as a file not existing.	Essential	Proficient	
Debug a program.	Based on a specification, hypothesize what cases are likely to expose faults in an implementation and write automated tests to exercise them.	Essential	Proficient	
Debug a program.	Based on an existing unit of a program, hypothesize what faults it contains and write automated tests designed to trigger them.	Essential	Proficient	
Debug a program.	Perform test-driven development by writing tests for requirements before modifying a program to satisfy those requirements.	Elective	Exposure	
Represent the data from a real-world problem domain within a program.	Use provided Booleans and logical operators and relations.	Essential	Proficient	
Represent the data from a real-world problem domain within a program.	Use provided numbers (including naturals, integers, and rationals), arithmetic operators, and ordering relations.	Essential	Proficient	
Represent the data from a real-world problem domain within a program.	Use provided product datatypes, such as record, struct, tuple, named tuple, object attributes, etc.	Essential	Proficient	

Category	Outcome	Importance	Proficiency	Dissent
Represent the data from a real-world problem domain within a program.	Use provided sum datatypes, such as enum, union, optional/maybe, etc.	Elective	Competent	Not covered at EOU.
Represent the data from a real-world problem domain within a program.	Use provided text datatypes (i.e. strings).	Essential	Proficient	
Represent the data from a real-world problem domain within a program.	Write programs that correctly handle character sets, text encoding, internationalization, and localization.	Elective	Exposure	Not covered at EOU.
Represent the data from a real-world problem domain within a program.	Use provided sequential data structures, such as arrays, including multidimensional arrays	Essential	Proficient	
Represent the data from a real-world problem domain within a program.	Use provided mapping data structures, such as maps/dictionaries	Essential	Competent	
Connect a program with its environment.	Read and write data on standard streams such as a console.	Essential	Proficient	
Connect a program with its environment.	Read and write text files in external storage. Understand the concept of working directory.	Essential	Proficient	
Connect a program with its environment.	Detect and handle exceptional conditions (e.g. missing file, invalid input) within a program (using return value, errno, exceptions, etc.).	Essential	Competent	
Apply structured programming principles.	Decompose a problem into blocks of sequential code, functions, and procedures	Essential	Proficient	
Apply structured programming principles.	Describe a function or block of code in terms of pre- and post-conditions.	Essential	Exposure	
Apply structured programming principles.	Recognize repetition in a codebase and identify how to refactor the instances into calls to a common function.	Essential	Proficient	
Apply structured programming principles.	Distinguish statements and expressions	Essential	Proficient	
Apply structured programming principles.	Use provided conditional/selection statements.	Essential	Proficient	
Apply structured programming principles.	Use provided iteration/repetition/looping statements.	Essential	Proficient	
Apply structured programming principles.	Implement and trace recursive solutions to problems	Essential	Proficient	
Apply structured programming principles.	Locate and apply existing libraries and algorithms to solve problems, making use of built-in functions and data structures when appropriate.	Essential	Exposure	
Apply object-oriented programming principles.	Decompose a program into objects that encapsulate state and behavior.	Essential	Competent	
Apply object-oriented programming principles.	Connect objects using composition and messages/methods, while using provided features to hide internal details of data representation.	Essential	Exposure	
Apply object-oriented programming principles.	Extend and reuse parts of a program, using provided features for inheritance, polymorphism, and genericity.	Essential	Exposure	Not covered at PSU
Apply object-oriented programming principles.	Understand the complexities associated with multiple inheritance. It is not assumed students will know techniques for doing it other than possibly via idioms like interfaces or mixins.	Essential	Exposure	Not covered at EOU. Not covered at PSU.
Apply object-oriented programming principles.	Identify how abstract base classes and interfaces differ from normal base classes. Use them appropriately in designing a simple inheritance heirarchy.	Essential	Exposure	Not covered at PSU
Apply object-oriented programming principles.	Distinguish static and dynamic dispatch, use provided features for both, and justify deciding between them.	Essential	Exposure	Not covered at EOU. Not covered at PSU
Apply object-oriented programming principles.	Use libraries written using object-oriented programming in a correct, idiomatic way.	Essential	Competent	Not expected by PSU

Category	Outcome	Importance	Proficiency	Dissent
Generalize understanding of programming.	Compare how a programming language feature is used in at least two languages that have it.	Essential	Competent	

Category	Outcome	Importance	Proficiency	BigO Expectations	Dissent
Abstract Analysis	Classify and rank common algorithms and data structure operations by asymptotic time complexity.	Essential	Competent		
Abstract Analysis	Understand the different cases for which complexity can be analyzed (best, worst, average) including basic amortized analysis (insertEnd in an array based list).	Essential	Competent		EOU only covers BigO
Abstract Analysis	Algebraically manipulate standard notations for complexity. (Recognize that n(3n + 2) => 3n^2 + 2n => O(n^2)).	Essential	Competent		
Abstract Analysis	Identify the asymptotic complexity of new algorithms involving standard operations on array, linked list, binary search tree, and hash table. Select appropriate tools based on time complexity	Essential	Proficient		
Abstract Analysis	Correctly identify the resources/operations to be counted in complexity analysis. e. g. Given a list of 10,000 numbers, but only 500 unique values, correctly reason about the efficency of building a set.	Essential	Competent		
Abstract Analysis	Describe the situations where asymptotic time complexity fails to capture important differences in algorithms. e.g. Small problem sizes and large constant factors (especially for algorithms in the same BigO category)	Essential	Competent		
Abstract Analysis	Discuss considerations other than time complexity that might be used to select an algorithm: space complexity, programming time, maintainability, etc...	Essential	Exposure		EOU saves for Algorithms
Searching & Sorting	Implement linear and binary searches	Essential	Proficient		
Searching & Sorting	Describe different characteristics for sorting algorithms including stability, in-place, adaptivity, partial sorting. Select appropriate algorithms based on these criteria	Essential	Proficient		
Searching & Sorting	Implement quadratic sorts - selection, insertion.	Essential	Proficient	Proficient	
Searching & Sorting	Implement mergesort and quicksort.	Essential	Proficient	Proficient	Not covered at EOU, UO
Searching & Sorting	Describe heapsort (Essential) and implement it (Recommended)	Essential	Exposure		Not covered at EOU
Searching & Sorting	Describe the logic of hybrid sorting algorithms (introsort, Timsort)	Recommended	Exposure		Not covered at EOU
Searching & Sorting	Describe non-comparison based sorting algorithms (bucket sort/radix sort) and identify situations in which they are appropriate	Recommended	Exposure		Not covered at EOU
General Data Structures	Recognize the difference between an Abstract Data Type and an implementation of that ADT.	Essential	Proficient		
General Data Structures	Describe expected operations for the following Abstract Data Types: list, sorted list, stack, queue, set, table.	Essential	Proficient		
General Data Structures	Describe expected operations for the following Abstract Data Types: deque.	Recommended	Proficient		
General Data Structures	Reason about different implementations of an abstract data type. Identify the contraints of implementations and recognize implicit vs explicit structure (representing a heap with an array; maintaining a sorted list).	Essential	Competence		
General Data Structures	Implement static or dynamically sized containers. Reason about complexities related to resizing a container and possible optimizations for a static container.	Essential	Competence	Competence	
General Data Structures	Build nested data structures using arrays and lists. (e.g. array of lists that might be used to implement a hash table)	Essential	Competence	Exposure	Not covered at EOU
General Data Structures	Implement shallow or deep copies of data structures and choose the appropriate type of copy for a particular job.	Essential	Proficient		
General Data Structures	Use features of a language to create a generic data structure. (i.e. a LinkedList that can hold any data type, not just a IntegerLinkedList)	Essential	Competence		
General Data Structures	Recognize iterator concepts and how to use them to write algorithms that interact with data structures.	Recommended	Exposure		Not covered at EOU
Lists	Implement a singly linked or doubly linked list including circular lists. For those structures implement fundamental algorithms like insert, remove, traverse, delete, and merge.	Essential	Proficient	Proficient	
Lists	Implement an array based list with fundamental algorithms like insert, remove, traverse, delete	Essential	Proficient	Proficient	
Stacks & Queues	Implement both stacks and queues using linked lists and arrays.	Essential	Proficient	Proficient	
Stacks & Queues	Use stacks and queues to implement algorithms. (e.g. basic parsing task using a stack)	Essential	Proficient	Proficient	
BST	Implement a node-based BST and fundamental algorithms like insert, contains, delete.	Essential	Proficient	Proficient	Not covered at UO
BST	Use appropriate terminology to describe BSTs and their nodes (height, depth, completness, subtree, etc...)	Essential	Proficient		Not covered at UO
BST	Implement pre/in/post order traversals and pick the appropriate strategy for a given task.	Essential	Proficient	Proficient	Not covered at UO
Heaps	Implement a binary heap with fundamental operations like add, get min/max, delete.	Essential	Competence	Competence	
Hashing & Hash Tables	Implement hash tables with fundamental operations like insert, remove, delete.	Essential	Proficient	Proficient	
Hashing & Hash Tables	Recognize what makes a good (or perfect) hash function. Construct a hash function for different data types.	Essential	Competence		
Hashing & Hash Tables	Use either open addressing or chaining to resolve colisions in a hash table.	Essential	Proficient	Proficient	
Self Balancing Trees	Describe how a self balancing tree will handle inserting or removing a value. Be familiar with logic behind AVL and/or RedBlack trees.	Essential	Competence	Competence	Not covered at UO
Self Balancing Trees	Describe how a B tree is organized and how values would be inserted or removed.	Essential	Competence		Not covered at EOU, UO
Graphs	Use appropriate terminology to describe graphs.	Essential	Competence	Not Expected	EOU covers in Algorithms.
Graphs	Identify different representations of a graph (adjacency list/matrix) and translate between them.	Essential	Proficient		EOU covers in Algorithms. WOU in Algorithms or math courses
Graphs	Identify traversal order for breadth first or or depth first searches on a graph.	Essential	Proficient		EOU covers in Algorithms. WOU in Algorithms or math courses
Graphs	Perform by hand basic graph based algorithms (e.g. shortest path, minimal spanning tree)	Recommended	Exposure	Not Expected	EOU covers in Algorithms. WOU in Algorithms or math courses

Category	Outcome	Importance	Proficiency	Command Line Expectations	Dissent
Power use of an operating system	Understand and proficiently navigate a file system including dealing with complexitites like hidden items and file extensions and basic permissions.	Essential	Competent	Compentent	
Power use of an operating system	Obtain, install, and configure needed tools in your native OS	Essential	Proficient		
Connect to other systems	Use ssh to open a terminal on a remote or virtual machine.	Recommended	Competent	Competent	Not at WOU. Essential at PSU
Connect to other systems	Understand the difference between local and remote resources and manage files on a remote system. This includes making use of locations like a remote server, in a cloud based service, or a local VM.	Recommended	Exposure		
Development tools	Execute a program using interpreter or compiler.	Essential	Proficient	Compentent	
Development tools	Run a program under a debugger to observe computations and isolate unexpected behavior.	Essential	Proficient		
Development tools	Use static analysis and/or dynamic analysis tools such as linters and memory checkers (as appropriate for language).	Essential	Exposure		
Development tools	Use a unit test framework to write/run automated tests.	Essential	Exposure		
Development tools	Use language reference documentation, tutorials, and other resources to figure out new features of a language and to learn new languages.	Essential	Competent		
Source control	Use git or a similar tool to perform basic operations (init, commit, checkout) on a code repository.	Essential	Competent	Exposure	
Source control	Use github or other online code collaboration tools to obtain assignment starters (cloning, pulling) and submit finished work (push). Understanding appropriate use in an academic context.	Essential	Competent		