TCSS 342 Master Syllabus
Version: April 2011
(Approved: 27 May 2011)

Catalog Description

Covers abstract data types, design and complexity analysis of data, and usage of generic data structure libraries in high-level programming languages. Includes sequential and linked lists, binary trees and balanced binary trees, B-trees, hash tables, and heaps. Prerequisite: TCSS 305; a minimum grade of 2.0 in TCSS 321.

Preconditions

- Recognize and use mathematical formalisms (e.g., sets, logic, summations, proof).
- Translate problem descriptions into mathematical formalisms.
- Design, implement, document and debug a medium complexity program with an object hierarchy that includes interfaces and/or abstract classes and a graphical user interface, given some guidance on the design.
- Design and implement unit tests for a medium complexity program with an object hierarchy that includes interfaces and/or abstract classes.
- Utilize modern software engineering tools (e.g., IDEs, static checkers, unit testing frameworks, revision control systems) during the implementation of a medium complexity program.
- Correctly employ programming language features by reading and interpreting the associated published API documentation.

Student Learning Goals (to be added to syllabus handed out to students)

- Define what a data abstraction is.
- Identify the operations that characterize the following abstractions and structures. These will be called the canonical set.
  Data abstractions: **List**, **Stack**, **Queue**, **Priority Queue**, **Map/Dictionary**
  Data structures: **Array**, **Linked List**, **Tree**, **Binary Search Tree** (BST), **Balanced BST**, **B-tree**, **Heap**, **Hash Table**
- Identify the API of each abstraction or its closest correlates in the canonical set as realized in a standard generic library of a modern programming language, such as the Java Collections Framework.
- Apply pictorial representations to explain the implementations of the data structures and their operations.
- Adapt and/or extend the generic code implementation of at least 3 data structures from the canonical set, at least 1 of which is recursively defined.
- Instantiate and apply the API's of at least 4 data abstractions (or their correlates) in the canonical set as provided in the standard generic library of a modern programming language.

- Analyze the worst- and average-case time and space complexity of the operations in common implementations of the data abstractions and structures in the canonical set.
- Analyze the worst-case time complexity of code that uses data abstractions in the canonical set.
- Compare and contrast time/space characteristics of different implementations of the same data abstraction.
- Select data abstractions, structures, and implementations that a developer would use in solving larger problems and defend the appropriateness of these choices.

CSS Degree Student Learning Outcomes that this course contributes to (to be added to syllabus handed out to students)

a. an ability to apply knowledge of computing and mathematics appropriate to the discipline;
b. an ability to analyze a problem, identify and define the computing requirements appropriate to its solution;
c. an ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs;
i. an ability to use current techniques, skills, and tools necessary for computing practice.

UWT Student Learning Goals that this course contributes to (to be added to syllabus handed out to students)

Inquiry and Critical Thinking
Students will acquire skills and familiarity with modes of inquiry and examination from diverse disciplinary perspectives, enabling them to access, interpret, analyze, quantitatively reason, and synthesize information critically.

Topics covered

- Review of operations on arrays and on linked structures
- Review of analysis of algorithms
- Implementation of generic classes and methods
- Review of linear data structures (Lists, Sets, Stacks, and Queues)
- Trees (Heap, BST, Balanced BST (AVL), B-Tree) – others as time allows (Red/Black, Splay, etc.)
- Map ADT and implementations (Hash tables)
- (Optionally, as time allows) Sorting algorithms
- (Optionally, as time allows) Introduction to the Graph ADT

Additional Information

The TCSS 342 course serves multiple purposes in both the BS and BA degrees in CSS. It is a listed pre-requisite for both TCSS 343 and 360, as well as for a large number of CSS concentration courses (i.e. electives in the major). The broad goals of TCSS 342 are for students to understand the concepts of a data abstraction (i.e. *abstract data type*), the operations associated with each of a set of canonical data

abstractions, the standard ways in which these abstractions are implemented using both the primitives and other abstractions in modern programming languages, how to use this canonical set as realized in the standard, generic libraries of modern programming languages, the analytical skill of worst- (big O) and average-case space/time complexity analysis, and the use of space/time complexity analysis in making choices between data abstractions and data structures for specific problems.

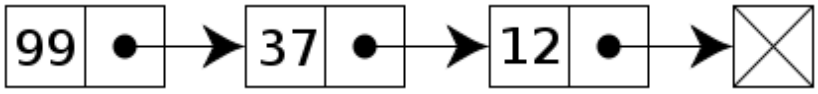
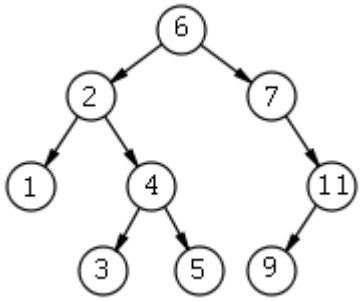
There is an optional 2 credit seminar workshop (TCSS 390C) associated with this course.

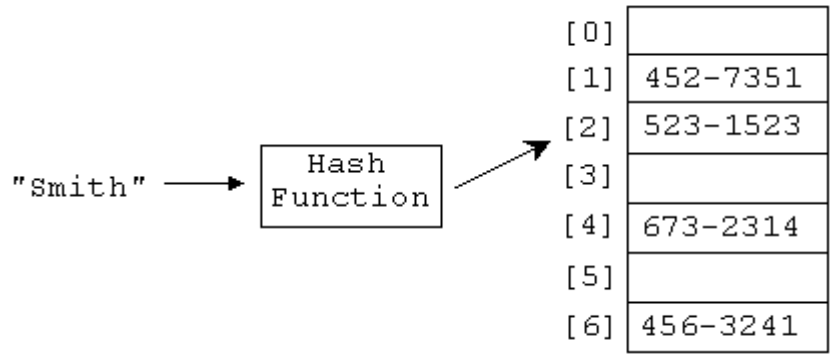
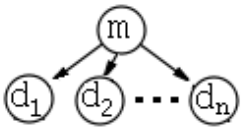
Recent textbooks used include:

Structures in Java- From Abstract Data Types to the Java Collections Framework, Simon Gray, 2007 (ISBN 0321392795)

Data Structures and Problem Solving Using Java Fourth Edition, Mark Allen Weiss, Addison Wesley, ISBN-10: 0-321-54140-5

The following shows diagrams typical of the pictorial representations referred to in the 4th Student Learning Goal:

Data Structure	**Pictorial Representation**
Linked List, Array	Linked List, Array 
BST, Balanced BST, B-tree **Data Structure**	Rooted Tree  **Pictorial Representation**
Hash Table	Table

	
Heap	Array, Partially Ordered Tree  $m < d_i$