Syllabus – CS 2402 Data Structures

• **Instructor:** Dr. Eric A. Freudenthal
• **Office:** CS 202a
• **Phone/e-mail:** 747-6954 efreudenthal@utep.edu
• **Text Book:** Alan Siegel's text, available from the copy center in the Union building

**Course Description:** The definition and implementation of abstract data types; representation of data using sets, lists, trees and graphs; the design and implementation of traversal, search, and sort algorithms; and the space and time analysis of algorithms. **Prerequisites:** CS 2401 and MATH 2300, each with a grade of “C” or better.

**Course Objectives:** The course will provide students with working knowledge of
- abstract data types,
- the representation of data as sets, lists, trees and graphs
- storage allocation and collection techniques, and
- basic algorithms and characterization of their expected & worst-case performance.

**Relationship of Course Lectures and Lab Sessions**
- Course lectures will discuss theoretical background. The structure of algorithms presented in this forum will be optimized to facilitate understanding of key concepts.
- Lab classes will be taught by TAs. The principal purposes of the lab assignments and TA sessions are to:
  - provide experience programming, debugging, evaluating, and using important algorithms and data structures
  - provide guidance in developing professional programming styles

**Assignments:**
- **Due Dates** Approximately five programming assignments will be given in the course. Completed assignments must be "handed in" to the teaching assistant on or before the due date in order to receive full credit. Assignments must be turned in at the beginning of class (within five minutes of the start of class). Absolutely no late programming assignments will be accepted beyond 1 week of the due date; late assignments will be penalized for lateness.
- **Grading:** Your program must implement the specified algorithms, and produce output as specified in the assignment. Programming assignments are graded on the following criteria
  - Correctness: Each program will be checked against a test suite constructed by the teaching assistant and professor. You are required to submit the test cases that you used to test your program. Note that there are several approaches to testing.
    - Code review, *i.e.* examination of the code and its documentation with the goal of detecting misinterpretations, inconsistencies, and specific faults, is typically the most effective.
    - Black-box testing, *i.e.* testing based on the specification, uses boundary value analysis to define and test from equivalence classes of inputs. If a test case that targets an equivalence class
works properly, then it is assumed that other cases from the same class will work properly. It is impractical to test all combinations of variable values.

- White-box testing uses information about the internal structure of the code. There are numerous strategies for white-box testing, including the following: statement testing (every statement is executed at least once in some test), branch testing (each branch of every decision point in the code is chosen at least once in some test), path testing (every distinct path through the code is executed at least once in some test), and definition-use path testing (every path from every definition of every variable to every use of that definition is exercised in some test).

You should use a combination of the strategies discussed here.

- Efficiency. You will be asked to analyze and note the space and time complexity of the code that you submit for grading.
- Documentation and programming style. Your code should be documented and written in a professional, consistent, understandable style. Every assignment should include a "README" document or web hierarchy that the TA can easily use to determine
  - How to compile & run your program
  - The general structure and any non-obvious aspects of your program.
  - The testing strategy utilized for your program
  - Any deficits in your program (you will lose less credit if a problem is documented than if it is discovered by a TA.)