Last updated 03 January, 2020

CS260 – Data Structures
Winter 2020, CRN: 34401, Credits: 4

Instructor: Joseph Jess

Web: http://cf.linnbenton.edu/bcs/cs/jessj/web.cfm?pgID=5404

email: jessj@linnbenton.edu

Initial class note: We are *not* in an introduction course to computer science (CS) anymore, and there will be topics that can be quite tough to understand at first exposure; this is perfectly normal! It may take a few looks at a structure or algorithm, potentially from multiple presenters, for all of its components to start making sense. We continue to add to the design, testing, and implementation concepts covered in earlier classes that use a programming language as a learning tool. We should expect to do much reading and research, much practicing, and much discussion of the topics we cover in class and in the coursework.

1. LBCC catalog course description, including pre-requisites/co-requisites:

This course explores the correct use of a variety of data structures in object-oriented programs. The topics covered include the uses of complexity analysis, simple and complex sorting algorithms, stacks, queues, priority queues, arrays, linked-lists, file processing, tree structures, binary search trees, hashing algorithms, heaps and recursion.

Prerequisite: CS 162 Introduction to Computer Science II with a grade of "C" or better.

2. Class Time-space:

2.1 Lecture + demo + lab: TR 1000 – 1220, MKH101

3. Measurable student learning outcomes:

At the completion of the course, students will be able to:

3.1 Demonstrate an understanding of complexity analysis and big O notation.
3.2 Analyze and write programming code for numerous searching and sorting algorithms.
3.3 Write programming code for linked lists and binary search trees.
3.4 Demonstrate an ability to trace code for sorting algorithms using recursion.
3.5 Demonstrate an understanding of hash tables, binary trees and multi-way trees.

I also hope to get practice with the following:

3.6 Writing algorithms using pseudocode.
3.7 Writing programs using stacks, queues, and graphs.
3.8 Demonstrate concepts related to file processing and storage management.

4. Learning resources:

4.1 **Note:** All class materials and storage will be freely available in a digital format
1.1 **The C++ language** – available through several media. We will discuss this quite a lot in class.
4.2 **A C++ compiler** – I will likely use GCC's gpp. We will of course talk about this more in class.
4.2.1 I might recommend the IDE called VSCode, I will likely just use a text editor and the command line to compile; most others would be fine.
4.2.2 We will discuss some capabilities of smart code editors during the course.
4.3 (**strongly recommended**) A desire to learn, experiment, design, test, and problem solve with code (both on and off of a computer).

5. Grading:

5.1 Scores for coursework items will be initially available when demonstrated to the instructor. We will keep a spreadsheet in a shared folder through our student email account using Google Drive. We will discuss this some in class, including how to access it and keep yourself organized (which may affect your grade).
5.1.1 My favored grading method is to have people show me where they are throughout the week so we can discuss approaches and any requirements that would affect the grade, which also allows for better understanding of requirements.

5.2 We will be required to turn in all coursework items before 23:59 (Pacific Time Zone) on the date that they are due (generally the first day of class each week in my courses).

5.2.1 We need to be sure to give ourselves plenty of time to submit coursework, as late work will not be accepted without prior consent or special circumstances.

5.3 To earn a passing grade in this course someone must pass each of the following coursework categories:

5.3.1 **Demonstration**: Discussion and weekly assignments – 50%

5.3.1.1 There are a number of weekly assignments to be completed for this class, designed to challenge and solidify design, coding, and testing skills.

5.3.1.2 Project components are generally graded based on:

5.3.1.2.1 completeness (does it compile and run)

5.3.1.2.2 correctness (does it meet the listed requirements)

5.3.1.2.3 quality and explanation of the design (features in advance, organized)

5.3.1.2.4 quality and explanation of the tests (test each feature and expected successes and failures)

5.3.1.2.5 quality of the implementation (consistent and readable style, runs well, is easy to learn and use)

5.3.1.2.6 **Note**: careful design, systematic testing, consistent style, and readability of code are important software quality factors (all of which are subject to interpretation but graded by the instructor based on the spirit and letter of the requirements, so be sure to explain your decisions).

5.3.1.2.7 **Note well**: Our submission needs to be understandable and able to function from just the files in the final submission. This means that everyone needs to include any and all files necessary for your project to function, even if the instructor provided them.

5.3.1.2.8 **Note very well**: Graded work must be designed and implemented by the student submitting the work and must function on one of the instructor's machines in order to get a final grade. (to create a working project quickly: get it working simply, then add to it; if at some point it stops functioning, then we will better know where an error was introduced)

5.3.2 **Final**: Final project – 50%

5.3.2.1 There will be a final project to test the overall ability to understand, design, implement, test, and reflect on the problem solving and coding knowledge and skills covered in the class.

5.3.2.2 The final project will be a mix of in-class (initial design, testing, and implementation discussion) and take-home (final design, testing, and implementation) elements.

5.3.3 Final grades will be given out based on the following based on score in the class:

90-100%: A
80-89%: B
70-79%: C
60-69%: D
00-59%: F

5.4 Reminder: A passing grade in order to count for course requirements for CS classes is generally a C or above.

6. Other Administrative Information:

6.1 For a list of general administration information (note that this list is not intended to be exhaustive), such as:

6.1.1 contacting me,
6.1.2 accessibility resources,
6.1.3 expectations of student conduct,
6.1.4 communications,
6.1.5 student assistance,
6.1.6 miscellany,
6.1.7 nondiscrimination & nonharrasment,
(each section contains a number of sub-sections and is not meant to be exhaustive of all situations)

see my administrative information document: *administrative_information* document.