COMP 262: Data Structures & Algorithms
fall 2017 Syllabus
Linfield College, Computer Science Department
Prof. Daniel Ford

"When you understand the cause, the solution is straightforward."

COURSE DESCRIPTION AND OBJECTIVES:
Adds data abstraction, intermediate data types and related algorithms to the beginning programming techniques learned in COMP 161. Lectures and mandatory one hour lab session per week.

PREREQUISITES:
Completion of COMP 161 and concurrent enrollment in MATH 230 or instructor approval is required. A 'B' or better in COMP 161 is recommended. We assume that you are comfortable writing, analyzing, and debugging programs in an object-oriented programming language, e.g. Java.

CREDITS: 3

PROFESSOR: Daniel Ford
Office: Renshaw 210 E-mail: dford@linfield.edu Cell: 503-857-2995
Office Hours: MTWThF 12:15—1:00 PM, and by appointment.

CS SYSTEM ADMINISTRATOR: Francisco Mora, fmora@linfield.edu.

LECTURES: Renshaw 211
Section 1: MWF 9:00 – 9:50 AM

LABS/WORKSHOPS: Renshaw 211
Wednesday 3—5 PM
Help and tutoring is also available through Linfield's Learning Support Center (http://www.linfield.edu/learning-support.html.)

SATURDAY ICPC PROGRAMMING CONTESTS / FIELD TRIPS:
October 7th 9:00 AM – 2:00 PM. ICPC North American Qualification Contest (Renshaw 211)
November 4th or 11th 9:30 AM – 8:00 PM. Pacific NW Region ICPC (George Fox University)

TEXT BOOK:
- **Thomas Cormen, Charles Leiserson, Ronald Rivest & Clifford Stein,** Introduction to Algorithms (3nd Edition) http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=11866, 2009, MIT Press **ISBN** 978-0-262-03384-8

OTHER RECOMMENDED BOOKS
Algorithms
- **Steven S. Skiena**, The Algorithm Design Manual (2nd Edition) $72 http://www.algorist.com/, 2008 Springer. **ISBN:** 978-1848000698.

C++
- **Mark Allen Weiss**, C++ for Java™ Programmers. $55. 2003, Prentice Hall **ISBN** 978-0139194245

Programming Competitions
- **Steven Halim & Felix Halim,** Competitive Programming, $18, http://www.lulu.com/product/paperback/competitive-programming/12110025.

RECOMMENDED (FREE) WEB RESOURCES:
Algorithms
- **Robert Sedgewick & Kevin Wayne**, The 50 algorithms that every programmer should know, http://algs4.cs.princeton.edu/code/
- **TopCoder Algorithm Tutorials** http://community.topcoder.com/tc?module=Static&d1=tutorials&d2=alg_index

- **USA Computing Olympiad (USACO) Training Program Gateway** train.usaco.org http://cerberus.delos.com:791/usacogate
- **Linfield Domserver** https://domserver.cs.linfield.edu/public/

C++

- **Cay Horstmann** Moving from Java to C++, http://www.horstmann.com/ccj2/ccjapp3.html
- **Cay Horstmann,** A Crash Course from C++ to Java, http://www.horstmann.com/ccc/c_to_java.pdf.
- **Beck Hasti** C++ for Java Programmers, http://pages.cs.wisc.edu/~hasti/cs368/CppTutorial/
- **Wikipedia** Comparison of Java and C++ http://en.wikipedia.org/wiki/Comparison_of_Java_and_C%2B%2B
- **David Flanagan,** How Java Differs from C http://oreilly.com/catalog/javanut/excerpt/index.html
- **Marvin Solomon,** Java for C++ Programmers..
  Version 1 adapted by Marius Zimand::http://www.ida.liu.se/~TDDI48/Java-vs-C++.html.
  Version 2 adapted by Vicki Allan:http://digital.cs.usu.edu/~allanv/cs4700/Java/java-tutorial.html.
- **Daniel Graziotin** Object-Oriented Memory Management (Java and C++) http://task3.cc/wp-content/uploads/2011/04/object-oriented-memory-management-java-c++.pdf

Interview Questions

- CareerCup http://www.careercup.com/

COURSE OBJECTIVES:

Upon completion of this course the student should understand the basic algorithmic factors affecting computational speed and be able to

- Practice learning a new programming language
- Practice writing, testing, and debugging programs in C++.
- Practice gathering and deciphering program requirements.
- Practice decomposing and organizing a problem into cohesive subtasks.
- Practice identifying and using appropriate data abstractions and algorithms.
- Practice writing, tracing and debugging algorithms.
- Practice searching and sorting.

GRADES:

Grading for this course will be based on:

homework	~35%
lab quizzes	~35%
final exam	~20%
ICPC Programming Contests	~10%

HOMEWORK:

Homework will be due Sunday at Midnight.
The solution to every problem must be written in both C++ and pseudocode. In addition for each problem you need to show your work, e.g.

- Summarize the problem
- Break the problem down into clearly defined subtasks/methods
- Explain the heart of the problem – the difficult part(s)
- Identify what data you need to solve the problem, which data structures you will use to store the data, and, if important, why you chose them.
- Show the results of tracing your algorithm with some simple inputs.
- Prove that your algorithm is correct. (Identify and prove all loop invariants.)
- Specify and/or log your testing and debugging sequence. (Use the strategy to "solve a simpler/partial problem first" and then progressively solve and debug more and more complicated problems.)
- Identify significant and/or interesting problems you encountered and what you learned.

LAB QUIZZES

This course includes a mandatory two-hour lab every Wednesday from 3 PM – 5 PM in Renshaw 211. These quizzes will generally focus on understanding, and partially or completely solving contest questions related to recent homework problems.

ICPC PROGRAMMING CONTESTS
The ICPC North American Qualification Contest is a 5-hour individual contest from 9am to 2pm on October 7th. You may use any resource available before the beginning of the contest.

The Pacific NW Region ICPC is a 5-hour contest from 12pm to 5pm on either November 4th or 11th where groups of three students will compete together as a team. You may only use the printed resources you bring to the competition. (Everyone is expected to compile a paper copy of all your programs and related notes to bring with you.)

Your grade for these activities will be based on how well you demonstrate that you've learned as much as possible related to the contest questions. This includes understanding the problem requirements and difficulties and other related course objectives listed above as well as writing a C++ program that successfully solves the problem. In the group competition in November students will also evaluate the other members of the group.

ASK WHY
Learning why something works is much more powerful than learning what you need to do to get something to work. Whenever tempted to ask what you should do, try asking why something is happening instead. When you understand the cause, the solution is straightforward.

TEST YOUR ASSUMPTIONS
We answer questions either by asking someone, looking something up, e.g. via a book, API, google search, stackoverflow.com, or the scientific method, i.e. testing hypotheses. Of these, testing hypotheses is the most important and requires the most practice. Running a program is akin to running an experiment. Running frequent small programs/experiments that differ from another test in only one way helps us learn much more and learn much more easily than using a few, infrequent, large, radically different tests. This is closely related to the *Test Driven Development* practice we implemented by using JUnit testing in COMP 160 and COMP 161.

ACADEMIC CONDUCT
In this course we will adhere to the college policy on academic honesty, as published in the Linfield College Course Catalog. Please review this policy if you are not already familiar with it.
Plagiarism is lying in an attempt to get credit for someone else's work.

- Never include someone else's code in something you are turning in for credit.
- Obey the *30 second long-term memory rule*. We all get stuck now and again. If you get help on a homework problem don't copy and paste what they did. Instead take a minute to think about their code why it works or doesn't work and how you might improve on their idea before starting to write your own solution from scratch. If this solution doesn't work see if you can fix it on your own first, and then repeat this step as necessary.
- Do not use any code that you do not understand. The more complicated the code is and less you understand exactly how and why it does what it does, the more comments you need to include with your code to explain how it works.

In computer science you will earn credit by combining basic building blocks, e.g. the vocabulary of a specific language, common patterns, well known data structures, and algorithms, into expressions, segments of code, procedures, functions, and entire programs that solve a particular problem or accomplish a specific task. The basic building blocks are common knowledge and thus represent ideas that do not need to be cited. You are expected to generate, evaluate, and illustrate your own ideas on how these building blocks can be put together to solve particular problems. The building block combinations that you submit for credit will illustrate a combination of ideas that is similar to the combinations used by others in some respects albeit unique in some other important ways.

ILLNESS/MEDICAL POLICY
In the unfortunate event that you must miss class you are encouraged to contact your workshop partner to find out what we worked on in class and to pick up any materials that were handed out. In general, homework extensions and make-up exams will be granted as needed on an individual basis.

DISABILITY SUPPORT
Students with disabilities are protected by the Americans with Disabilities Act and Section 504 of the Rehabilitation Act. If you are a student with a disability and feel you may require academic accommodations contact Cheri White, Assistant Director of Learning Support Services (LSS), within the first two weeks of the semester to request accommodations. LSS is located in Walker 126 (503-883-2444). We also recommend students communicate with their faculty about their accommodations and any special needs an instructor should be aware of.