This is an archive from winter **2018**. For winter **2019**, please visit https://web.eecs.umich.edu/~nham/EECS398W19/

EECS 398 System Design in C++
Winter 2018 Syllabus

Instructor Nicole Hamilton
nham@umich.edu
https://web.eecs.umich.edu/~nham/
C: 425-765-9574

Lectures Dow 1014
Tuesdays and Thursdays, 3:00 pm to 5:00 pm

Office hours Beyster 2649
Mondays and Wednesdays, 4:30 pm to 6:00 pm (280 has priority)
Tuesdays and Thursdays, 5:00 pm to 6:00 pm (398 has priority)

Prerequisites EECS 280 and 281

Organization

This will be a course in how to tackle a large project in C++. Starting from scratch, you will build a whole working search engine in C++ as a vehicle for learning how developers ("devs" in industry parlance) work in teams to tackle a large software design problem they don't already know how to solve and how to write beautiful code you know will work.

Here are main learning objectives:

1. The knowledge, skills and ability to work on a team to research, define, estimate, plan and carry out the design of a large, complex software project, especially, one you don't already know how to do.
2. Learn the basics of how a search engine works.
3. An appreciation of software as art and of the software design process as a fundamentally creative activity where you make choices.
4. The ability to recognize, appreciate and most important, *create* elegant, reliable code in C++, making effective use of the language features.
5. An understanding of how a large system is decomposed into objects with interfaces exposing a limited surface area and how they talk to each other.
6. An appreciation of how a complex application interacts with the operating system, the file system and the user.
7. An ability to apply and use software metrics, e.g., LOC and defect rates.

The course has been approved as an upper-level elective for CS, a flex tech elective for EE, an EECS elective for CE and an advanced tech elective for DS.

The project

You will self-select into teams of 4 or 5 to design and build a very simple search engine and assign your own roles. Choose your teammates wisely. These are the basic pieces you will need.

1. HTML parser.
2. Crawler.
3. Index.
4. Constraint solver.
5. Query language.
6. Ranker.
7. Front end.

In lectures, we'll walk through the problem posed by each piece and how it might be solved. You will also find the text helpful. But there will be no "starter files". Since this isn't a turnkey student project that I've already coded myself and assigned many times, you will have to make many of your own decisions about what's minimally needed, what would be nice to have and how you'll get it done on time.

All your work will be in C++ 11.0 and all of it must be yours. I intend to discourage use of STL, deliberately forcing you to do all your own problem-solving. You will figure out how to write your own templates, basic data structures and simple algorithms and get them right every time. No matter what part of this engine you build or what you contribute to any future large software project, these are the basic skills that you will need. These are the skills you will practice here.

All your code must adhere to the style sheet I'll provide.

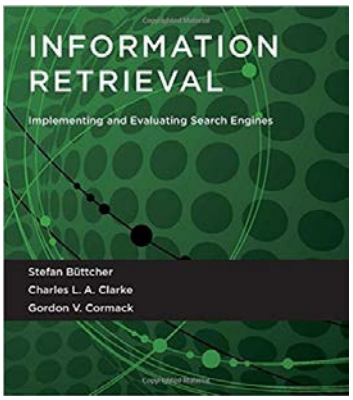
Your grade on the project will depend in roughly equal measure on your individual and team performances. Some of your grade will be determined on a competitive basis by ranking your engine's overall performance, features and quality against the engines created by the other teams. Your individual performance will be based primarily on the code you write, considering the number of lines of code you contribute to the project, the complexity of the tasks it performs, its performance, the creativity it displays, and the overall elegance.

Homework

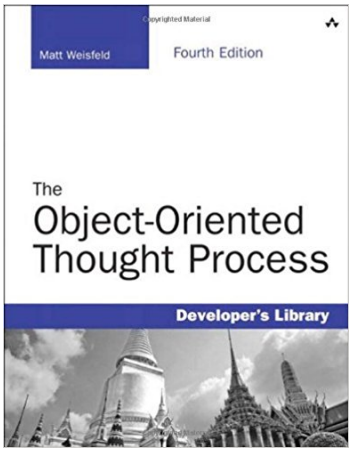
I anticipate perhaps a half-dozen homework assignments at two or three-week intervals.

Texts

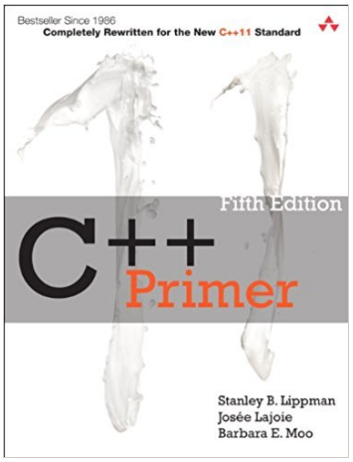
We will be using these texts. The first two are required. What I like about *Information Retrieval* is describes a lot of the pieces (just not ranking!) in the same way we thought about it at Microsoft.



Information Retrieval: Implementing and Evaluating Search Engines
Reprint edition (February 12, 2016)
Stefan Büttcher, Charles L.A. Clarke, Gordon V. Cormack
The MIT Press
ISBN 978-0262528870



The Object-Oriented Thought Process
Fourth edition (March 23, 2013)
Matt Weisfeld
Addison-Wesley Professional
ISBN 978-0321861276



C++ Primer
Fifth edition (August 16, 2012)
Stanley B. Lippman, Josée Lajoie, Barbara E. Moo
Addison-Wesley Professional
ISBN 978-0321714114

Grading

Here's a rough idea of the weighting I expect to apply.

Group project and other activities	
Group performance	30%
Individual contribution	30%
Homework	10%
Midterm	15%
Final	15%

I grade on the curve, scaling the results by deciding what I think was an A and what I think was a C and then interpolating in between. Most students will fall between 2.8 and 4.0 with median around 3.2 to perhaps 3.4.

Late policy

Late submissions will not be accepted.

Late withdrawals

I intend to deny virtually all requests for withdrawals once the teams are formed and underway.

Course revisions

This is the first time for this course and I'll be creating it on the fly. I reserve the right to make reasonable revisions to content, assignments, exams or grading as I decide may be appropriate.

All the work must be your own

You may certainly compare notes with other students and other teams and of course I understand that you may do research using Google. But absolutely everything you turn in to me must be your own work. On your exams, you will be required to copy the Honor Pledge in your own handwriting and sign your name to it.

> *I have neither given nor received unauthorized aid on this examination, nor have I concealed any violations of the Honor Code.*

I report everything.

Disability

Access and Accommodations: Your experience in this class is important to me and to the University of Michigan, where it is our policy and practice to create inclusive and accessible learning environments consistent with federal and state law.

If you experience barriers based on a temporary or permanent disability (including, but not limited to mental health, attention-related, learning, vision, hearing, physical or health impacts), please seek a meeting with the Services for Students with Disabilities (SSD) office at 734-763-3000 or https://ssd.umich.edu to help us determine appropriate academic accommodations. SSD typically recommends accommodations through a Verified Individualized Services and Accommodations (VISA) form. Any information you provide is private and confidential and will be treated as such.

If you have already established accommodations with SSD, please tell me what they are so I can be sure to provide them.

List of topics

Here is a rough outline of the topics we'll discuss, interwoven over the semester. We obviously won't dive into everything to the same depth. The C++ and OOP topics should already be pretty familiar from 280 and 281.

1. Introduction to the course. Expectations, grading, coding style requirements. Brief overview of the project and of how a search engine works.
2. The role of ideas and creativity in software design. Software as art. Beautiful code, efficiency, elegance. The style guide we'll follow.
3. The software design process.
    a. The initial vision. Something has become possible.
    b. The typical dev organization, devs, testers, program managers, leads and managers.
    c. Research, reverse engineering, brain-storming.
    d. Problem-choosing, deciding what's in and what's not.
    e. Decomposition into an architecture, a design strategy and a working definition of the interfaces.
    f. The product spec.
    g. Planning and estimating, KLOCs and other metrics, milestones.
    h. Problem-solving, coding and testing.
    i. Code freeze, triage, release.
4. Search engine basics.
    a. Brief history of information retrieval and web search.
    b. The concept of relevance.
    c. Dynamic and static rank.
    d. Text formats and code sets, Unicode and UTF-8.
    e. Simple HTML, title, heading, body text, links and anchor text.
    f. Tokenization, stop words, stemming.
    g. Crawling and the frontier.
    h. The inverted word index and the constraint solver.
    i. Query compiling, BNF grammars, TDRD parsers.
    j. Ranking and learning techniques.
    k. Snippets.
    l. Spam, link farms, traps, objectionable content.
5. C++ and the object-oriented thought process.
    a. Statements, expressions, primitive datatypes, arrays, pointers, references.
    b. Functions, passing by value and by reference.
    c. Structs, composition.
    d. Classes, inheritance, encapsulation, private versus public.
    e. Constructors and destructors.
    f. Virtual functions, abstract classes, polymorphism, UML.
    g. new and delete.
    h. Containers, iterators.
    i. Operator overloading.

j. try/throw/catch exception handling.
k. Templates.
l. Functors.
m. Lambda expressions.

6. Simple data structures and algorithms.

    a. Big O.
    b. Linked lists, queues, priority queues, hash tables, binary trees.
    c. Recursion, sorting and searching.
    d. Table-driven programming.
    e. Top-down recursive descent parsing.

7. OS facilities.

    a. Processes and threads.
    b. The file system.
    c. Pipes.
    d. Producer-consumer relationships.
    e. Shared memory.
    f. Memory-mapped files.
    g. Locks, critical sections, race conditions.