EECS 280 Winter 2016 Syllabus

Instructors

James Juett jjuett@umich.edu
Amir Kamil akamil@umich.edu

Contact

Please direct technical questions to our Piazza forums. For other questions, you can reach the course staff at eecs280staff@umich.edu .

Schedule

Mon, Wed	9:00 am - 10:30 am	AUD CHRYS	Prof. Juett
Mon, Wed	10:30 am - 12:00 noon	1013 Dow	Prof. Juett
Mon, Wed	12:00 noon - 1:30 pm	1013 Dow	Prof. Kamil
Mon, Wed	4:00 pm - 5:30 pm	182 Weiser	Prof. Kamil

For office hours and discussion sections, please refer to the calendar on the course website.

Textbook

Optional: C++ Primer, by Lippman, Lajoie and Moo. 5th edition.

Our textbook is available in print form, electronic Kindle edition, and for free through the University Library. If you decide to rely on the library version, please be aware that the number of simultaneous users is limited, and sometimes there may be a wait to access it.

Overview

This course provides a foundation for programming, the science of “how to” as a discipline. Students will learn a variety of techniques and principles to quickly write correct programs that are easy for others to understand and adapt to new purposes. This course focuses on

programming in the small, writing short programs or program components that are easily understood by a single programmer. The concepts covered in this course are not just about C++ programming, but we will see how they can be concretely realized in the C++ language.

By the end of this course, a successful student will be able to:

- Take a problem and consider various possible approaches for solving it
- Select an approach or algorithm that provides for a simple, clean, well-structured solution
- Convert the algorithm into C++ code, using good design and style
- Test and debug the program using rigorous techniques
- Understand the concepts of top-down design, data encapsulation, information hiding, and procedural and data abstraction
- Design, implement and use complete classes, including constructors, destructors, and operator overloading
- Implement dynamic data structures for stacks, queues and lists
- Be able to quickly design, implement, test and debug a large scale project independently (1000+ lines of code).

Topics

- Functional abstraction
    1. Specification
    2. Recursion
    3. The recursion/iteration duality
    4. Recursive/iterative invariants
    5. Functional generalization

- Data abstraction
    1. Formal notions of type and type hierarchies
    2. Simple and aggregate types
    3. Abstract Data Types: ADTs
    4. Abstraction functions and representation invariants
    5. Polymorphism

- Dynamic resource management
    1. Creation and destruction
    2. Rules of safety
    3. Container types
    4. Container iteration

In our exploration of these topics, we will discuss many elements of the C++ language, such as:

- Arrays (1-dimensional and multidimensional)
- File I/O

- Structs, classes, and objects
- Overloading of functions and operators
- Friends
- Strings (C-style and C++ string class)
- Pointers and dynamic arrays
- Templates
- Linked lists, stacks, and queues
- Iterators
- Exceptions
- Recursion
- Functors

Prerequisites

The prerequisite for EECS 280 is EECS 182 or EECS 183 or ENGR 101 or ENGR 151 or permission of instructor. The first project should be simple if you have satisfied the prerequisites.

Website

Most course materials are made available on the course website, and are considered required reading. A detailed schedule, including lecture topics, assignment due dates and exam dates, is also available on the course site.

Forum

We will be using Piazza to host a course forum. You are required to read this regularly; it is the venue we will use for important course announcements and project clarifications. In addition, it will be a significant source of help and hints on the projects.

We do not answer technical questions via email. In order to save everyone time, we want all students to have the benefit of seeing each question and its answer, so please use the forum.

We ask that you do not post your own solutions, project code, test cases, or output to the forum. Also, please search the forum before posting to avoid questions that have already been asked and answered.

Projects

Over the course of the semester, we will assign five programming projects. The first will be short and is based only on things we assume you have covered in prerequisites. The other projects

will be longer and cover major course topics.

Programming Environment

For this course, you must have a CAEN account. Anyone enrolled in EECS 280 is eligible for one, but it takes a little while to get it set up. We will grade your programs in the CAEN Linux environment and they must be submitted, compile, and run correctly in this environment. At the time of this writing, this is version 4.8.5 of the g++ compiler.

You are free to develop your programs on any platform you like, but you may use only ANSI/ISO standard C++, and are responsible for any differences between your preferred platform and the grading platform. To be safe, though, we recommend you develop in the CAEN environment, using the machine pool loginlinux.engin.umich.edu. The first discussion section covers different ways to connect to CAEN linux.

Successfully Completing the Projects

We have found through many years of teaching experience that the most common reason for poor project performance is **not starting early enough**. Plan to do some work on the project every day and try to have it finished a few days ahead of the due date, because many unexpected problems arise during testing. In addition, the computing sites can become very crowded, making it difficult to get a computer to use, so plan for these things to happen.

The second most common reason for not doing well on the projects is not asking for help when you need it. We offer help in office hours and on the class forums. When you come to office hours, please be ready to provide access to your code, preferably electronic. Another good way to get help is to post a question to the course forum. Remember, if you find that you are stuck on a piece of your project for an undue amount of time, please see us!

One major goal of this course is for you to learn to test and debug your programs independently. We will not debug them for you. Instead, we will help you try to figure out how to test and debug your program yourself. We will also ask you to demonstrate what testing and debugging techniques you have already tried, and what the results were, before providing any advice.

Finally, always make multiple backup copies of your work! If you somehow lose your work, it is your responsibility.

Project Evaluation and Grading

Each project will be evaluated for behavioral correctness, adherence to course principles, and general style. Some projects may also be evaluated on the quality of test cases you supply.

A program is correct if it behaves as specified in the project handout, for example, by generating the correct output. A program adheres to course principles when it uses techniques taught in lecture and asked for in the specification. For example, in Project 2, we will ask you to write

recursive functions. A function that uses a loop instead of recursion would not adhere to course principles, even if it produces the correct answer. Finally, a program should be readable by other programmers. We have posted general guidelines for good style on the course web site.

Due Dates and Extensions

Programs turned in after the exact time and date on the assignment will receive a zero. We do not generally offer extensions. For example, we do not offer extensions due to crowded computing sites, internet access problems, accidental erasure or loss of files, or outside conflicting commitments.

We will consider extension requests made in person and at least 2 weeks in advance, for example, for religious holidays. Additionally, we will consider requests for extensions due to documented, unanticipated medical or personal emergencies. If you can't see the instructor in advance due to the emergency, then see him/her as soon as you possibly can. In all cases, we require written proof of the emergency.

Academic Integrity

You must complete programming assignment 1 alone. You may complete programming assignments 2 - 5 either alone or with a partner. All programming assignments in this course are to be done by you or your partnership.

You may give or receive help on any of the concepts covered in lecture, discussion, or the textbook, and on the specifics of C/C++ syntax. You are allowed to consult with other students in the class to help you understand the assignment specification (the definition of the problem).

You may not collaborate in any way with people outside your partnership when constructing your solution; your partnership working alone must generate the solution to a programming assignment and must submit your code for grading together (i.e. your partnership may only submit one version of your code for grading). You are not allowed to work out the programming details of the problems with anyone outside your own partnership or to collaborate to the extent that your programs are identifiably similar. You may not derive your solution in any way from prior solutions. You may not share code outside of your partnership, including making it publicly available in any form (e.g. a public GitHub repository). You may **not** share test cases outside of your partnership as we consider your test cases part of your solution. If you have any questions as to what constitutes unacceptable collaboration, please talk to the instructor right away. You are expected to exercise reasonable precautions in protecting your own work. Do not let other students borrow your account or computer, do not leave your program in publicly accessible directory, and take care when discarding printouts.

We report suspected violations to the Engineering Honor Council. To identify violations, we use both manual inspection and automated software to compare present solutions with each other and with prior solutions. The Honor Council determines whether a violation of academic

standards has occurred, as well as any sanctions. Read the Honor Code for detailed definitions of cheating, plagiarism, and other forms of academic misconduct.

Guidelines for Partnerships

Working in a partnership is optional. For programming assignments 2 - 5, you may work either alone or in a partnership. The following guidelines apply to those who choose to work in a partnership.

If you work in a partnership, you and your partner will submit one assignment together. Write both of your uniqnames in the comments at the top of every file to avoid a potential referral to the Honor Council. You cannot change partners in the middle of one project, unless your partner drops the course. You may change partners only after a project is completed and submitted. However, you are free to work individually as much as you like or collaborate as much as you like, as long as it is with your partner.

DOs

- Do READ THESE GUIDELINES CAREFULLY before programming with another student. You must follow these guidelines, or risk being investigated for an Honor Code Violation.
- Do choose a partner from the current semester of this course.
- Do put both your uniqname and the uniqname of your partner in the comments at the top of all code files. This is important to avoid referral to the honor council.
- Do submit one copy of the project together.

DON'Ts

- Do not program with someone without understanding these guidelines.
- Do not partner on an assignment with someone who has already solved the problem. Students who do this will not learn as much as those who pair with someone at a similar skill level.
- Do not share code with anyone other than your partner, or a staff member.
- Do not split the work in half. It is important that both partners work on all parts of the program. Splitting the work may harm your or your partner's understanding of that part of the solution.
- Do not partner with anyone who is not currently enrolled in the course.

Discussion Sections

In discussion sections, you will gain practice with the material from lecture, with an emphasis on short programming tasks in small groups. The exercises assigned during discussion section are intended to be completed during discussion time. The work from discussion will be turned in, and graded for completeness. We encourage group work during discussion.

Grading and Exams

Your final grade is based on scores from programming projects, participation in discussion sections and two exams. The tentative point distribution is included in the following table. It is not likely that this will change, but circumstances might occur which would make changes necessary, at the discretion of the instructor.

Discussion exercises	5%
Programming projects (p1 4%, p2-p5 9%)	40%
Midterm exam	25%
Final exam	29%
Participation in course surveys, etc.	1%

There are no letter grades for individual projects or exams. The final course letter grade is based on the total weighted score earned. Final grades will be assigned based on the distribution of earned scores, consistent with past incarnations of EECS 280. In the past, the median student has received a grade in the low B range, and students within one standard deviation of average can generally expect to pass. We may adjust our distribution up or down if this semester's' group does particularly well or particularly poorly in comparison to past groups. In particular, if everyone does great work, we'll assign everyone a great grade! Please note, we do not offer the opportunity for "make-up" or extra credit work to improve your grade.

Exams

We expect you to take both exams at the scheduled times. If you miss an exam, and a medical or personal emergency is not involved, you will receive a zero for that exam. If you anticipate an exam in another course or a religious holiday which conflicts with our exam time, you must notify the instructor at least two weeks before the exam date. The exam dates are given at the beginning of the term so that you can avoid scheduling job interviews or other commitments on exam days; hence job interviews, etc. are not considered valid reasons for missing an exam.

Midterm: *Monday, 22 February 2016, 7:00pm - 8:30pm*
Final: *Thursday, 21 April 2016, 10:30am - 12:30pm*

Regrade Requests

While we work hard to grade accurately, but we sometimes make mistakes. If you believe we graded an assignment of yours incorrectly, you can submit a regrade request no later than one week after the graded work is originally returned. We will then re-grade your entire assignment,

which can cause your grade can go up, but it can also go down.

Submit project regrade requests by email to the instructor. Submit discussion regrade requests by email to your GSI. For exams, staple a written description of the grading error to the exam and give it to your instructor.

Tips for Doing Well in the Class

You will maximize your grade, and learn a lot at the same time, if you:

- Attend all lectures and discussion sections (note that many tips and hints about projects are given during class!)
- Read the assigned readings (textbooks, handouts, forum, web pages)
- Hand in your work on time (even if a program does not work, turn in whatever you have done)
- Start the programming projects early and come for help as soon as you need it
- Follow the program specifications carefully

Accommodations for Students with Disabilities

If you think you need an accommodation for a disability, please let your instructor know at the beginning of the semester or at least 2 weeks in advance. Some aspects of this course may be modified to facilitate your participation and progress. As soon as you make us aware of your needs, we can work with the Services for Students with Disabilities (SSD) office to help us determine appropriate academic accommodations. SSD (734-763-3000; http://ssd.umich.edu) 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.