CS 136: Elementary Algorithm Design and Data Abstraction

Official calendar entry: This course builds on the techniques and patterns learned in CS 135 while making the transition to use of an imperative language. It introduces the design and analysis of algorithms, the management of information, and the programming mechanisms and methodologies required in implementations. Topics discussed include iterative and recursive sorting algorithms; lists, stacks, queues, trees, and their application; abstract data types and their implementations.

Course Staff – Overview

Instructors:

- Adrian Reetz
- John Akinyemi

Other course personnel:

- ISAs (Instructional Support Assistants)
- IAs (Instructional Apprentices)
- ISC (Instructional Support Coordinator)
CS 136 – Main Topics & Themes

• imperative programming style
• elementary data structures & abstract data types
• modularization
• memory management & state
• introduction to algorithm design & efficiency
• designing “medium” sized, “real world” programs with I/O

Curriculum

Three of the most common programming paradigms are functional, imperative and object-oriented.

The first three CS courses at Waterloo use different paradigms to ensure you are “well rounded” for your upper year courses.

CS 135 ⇒ CS 136 ⇒ CS 246

functional imperative object-oriented

Each course incorporates a wide variety of CS topics and is much more than the paradigm taught.

Programming Languages

Most of this course is presented in the C programming language.

While time is spent learning some of the C syntax, this is not a “learn C” course.

We present C language features and syntax only as needed.

We occasionally use Racket to illustrate concepts and highlight the similarities (and differences) between the two languages.

What you learn in this course can be transferred to most languages.
Course Content – Overview

Traditionally, there are three content components to this course:

- **Lectures** for introducing new concepts
- **Tutorials** for (interactively) demonstrating the application of a new concept
- **Labs** and **Office hours** for one-on-one help with concepts or their application

Lectures

This term CS 136 follows a “flipped classroom”-approach. Instead of an instructor presenting course content in lectures, it is **your responsibility to read through the lecture notes yourself**.

Lecture notes:

- Available on the web page and as a printed course-pack from W Store | Course Materials + Supplies in SCH

Textbook:

- “C Programming: A Modern Approach” (CP:AMA) by K. N. King. *(strongly recommended)*

Lecture Notes

You will notice different styles of boxes throughout the notes:

- **Important information** appears in a thick red box.
- **Side notes** appear in a thin green box.
- **Advanced material** appears in a dashed yellow box. It provides deeper insights and might appear on assignments.
Lecture Schedule

This schedule is tentative, *i.e.*, it might change throughout the course.

- **Week 1** Wed, May 13 - End of Section 1
- **Week 2** Wed, May 20 - End of Section 2
- **Week 3** Wed, May 27 - End of Section 3
- **Week 4** Wed, Jun 3 - Section 4 – slide 52
- **Week 5** Wed, Jun 10 - Section 5 – slide 22

Check the CS 136 web page and Piazza regularly for updates on the lecture schedule!

Lecture Schedule

- **Week 6** Wed, Jun 17 - Section 6 – slide 34
- **Week 7** Wed, Jun 24 - Section 7 – slide 31
- **Week 8** Wed, Jul 1 - Section 8 – slide 16
- **Week 9** Wed, Jul 8 - Section 9 – slide 24
- **Week 10** Wed, Jul 15 - Section 10 – slide 15
- **Week 11** Wed, Jul 22 - Section 11 – slide 30
- **Week 12** Wed, Jul 29 - Section 12 – slide 17
- **Week 13** Wed, Aug 3 - End of Section 13

Lecture Q&As

Every week, the instructors provide scheduled Question & Answer sessions. Their purpose is answering questions related to the *scheduled lecture content*. These Q&As are open to any student and can be attended by multiple (100+) students at a time.

Q&A sessions are held on MS Teams (CS136-S20-Class)

The schedule is available on the CS 136 web page; we will send out reminders via Teams / Email before each session.
Lecture Q&As

Even if you don’t have a question, it might still be useful to tune in:

- listening to your colleagues’ questions allows you to compare your knowledge and assess your own progress
- listening to answers reinforces your knowledge
- being able to answer your colleagues’ questions is one of the best indicators that you’ve understood the lecture content

We expect that you have read through the scheduled lecture content beforehand and that you have established a basic understanding already. The Q&As are neither a substitution for a lecture nor for your own, independent learning effort.

Lecture Videos

On select topics we might provide additional instructional short videos (10 to 15 minutes) that outline, explain, or recapitulate particularly crucial or difficult concepts. These videos will be announced and made available on the course web page.

These videos are an addition to the lecture notes and the textbook. Solely relying on them won’t be enough to successfully complete CS 136.
Tutorials

Tutorials showcase the process from a (simple) problem statement to a complete working solution.

We will post tutorial videos weekly to the course web site. Depending on the complexity of the problem, tutorials can be between 15 and 90 minutes long.

We strongly recommend following along with tutorials as they teach you how to approach, analyze, and solve programming problems. They help you getting into the “programmer’s mindset”.

Labs & Office Hours

Both ISAa and instructors will provide scheduled one-on-one sessions every week. Feel free to ask any kind of questions; as a rule of thumb, ISAs focus more on assignments and programming problems, and instructors more on lecture notes and theoretical concepts.

Labs and office hours are held on MS Teams (CS136-S20-Class). The schedule is available on the CS 136 web page; we will send out reminders via Teams / Email before each session.

Leave a message for the ISA or instructor to get into the waiting queue; we will call you when it is your turn.

Marking scheme – Overview

- 90% assignments (roughly weekly)
- 10% quizzes
Assignments

Assignments are roughly weekly (approximately 10 per term).

Assignment weight will increase throughout the term:

- A0 does not count toward your final grade, but must be completed before you can receive any other assignment marks.
- A1 – A4: each worth 5% of your final grade
- A5 – A8: each worth 10% of your final grade
- A9 & A10: each worth 15% of your final grade

The Most Important Rules for Assignments

Read the assignment instructions carefully.

Read the official Piazza post and check back frequently for updates.

Remember that rules & requirements may change throughout the course.

Levels of Collaboration in Assignments

Assignment questions are individually colour-coded as either Black or Gold to indicate if any collaboration is permitted.
Black Assignment Questions

For Black questions, moderate collaboration is permitted:

- you can discuss assignment strategies openly
- you can search the Internet for strategies or code examples
- you can discuss your code with individuals, but not with a larger group
- you can show your code to others to help them (or to get help), but copying code is not allowed (electronic transfer, copying code from the screen, printouts, etc.)

If you submit any work that is not your own, you must cite the origin of the work in your source code. By not citing, you are presenting work of others as your own; this is called plagiarism and constitutes a violation of academic integrity.

Gold Assignment Questions

For Gold questions, no collaboration is permitted:

- never share or discuss your code with other students
- do not discuss assignment strategies with fellow students
- do not search the Internet for strategies or code examples

You may always discuss your code with course staff.

When asking on Piazza regarding a Gold question, your post must be private (Post to: Instructors).

Academic integrity is strictly enforced for Gold questions.
Hand-marking

Assignment questions might be labeled as hand-marked. They may be evaluated for style:

- documentation and comments
- code readability
- white space and indentation
- identifiers (variable & function names)
- appropriate use of helper functions
- testing methodology

The purpose of hand-marking is not to “punish” or “torture” you. It is formative feedback to improve your learning.

Unfortunately, we do not have the resources to hand-mark all assignment questions.

Well formatted and documented code is still expected, even if it is not hand-marked.

We will not provide assistance (office hours or piazza) if your code is poorly formatted or undocumented.

View your formative feedback on MarkUs.

Second Chances for Assignments

Assignment deadlines are strict, but some assignment questions may be granted a “second chance”.

- Second chances might be granted automatically, depending the quantity and quality of the submissions.
- Don’t ask in advance if a question will be granted a second chance; we won’t know.
- Second chances are (typically) due 48 hours after the original.
- Your grade is: \( \max(\text{original}, \frac{\text{original} + \text{second}}{2}) \) (i.e., there is no risk in submitting a second chance).
Assignment Implementation via Seashell

We use our own development environment called Seashell:

- browser-based for platform independence
- works with both C and Racket
- integrates with Marmoset, our submission & testing environment
- helps to facilitate your own testing

See the website and view Tutorial 01 for how to use Seashell.

Design recipe

In CS 135 you were encouraged to use the design recipe, which included: contracts, purpose statements, examples, tests, templates, and data definitions.

The design recipe has two main goals:

- to help you design new functions from scratch, and
- to aid communication by providing documentation.

In this course, you should already be comfortable designing functions, so we focus on communication (through documentation).

Documentation

In this course, every function you write must have:

- a purpose statement, and
- a contract (including a requires section if necessary)

Unless otherwise stated, you are not required to provide templates, data definitions, or examples.

Later, we extend contracts to include effects and time (speed / efficiency).
Assignment Submission via Marmoset

Assignments are submitted to the Marmoset submission system:
http://marmoset.student.cs.uwaterloo.ca/

There are two types of Marmoset tests:

- **Public** (basic / simple) test results are available immediately and ensure your program is “runnable”
- **Private** (comprehensive / correctness) test results are available after the deadline and fully assess your code

**Public tests do not thoroughly test your code.**

• Marmoset uses the best result from all your submissions, and we encourage frequent submission and re-submission

• for questions that are hand-marked, we mark the submission with the highest score; if two submission have the same score, we mark the one that was submitted closest to the deadline

• when you submit your assignments, you can view public test results immediately in Seashell

**To view your private test results, you must log into Marmoset after the deadline.**

Quizzes

We provide you with quizzes to encourage active learning and provide timely feedback that allows you to assess your understanding of the course content.

- new quizzes are posted every Monday on LEARN (learn.uwaterloo.ca/d2l/home/539714) and are due Sunday night
- these quizzes cover the week’s lecture slide and tutorial content

**The format and grading scheme can vary from week to week. Make sure to read the instructions carefully.**
Support

In case of a problem, stay calm and follow this list:

1. check CS 136 web page,
2. search Google (DuckDuckGo, etc.),
3. search on Piazza, and last
4. contact us.

Course announcements made on Piazza are mandatory reading (including official assignment and exam posts).

Contact us

To get in touch with us you can

• post on Piazza,
• file a Request,
• talk to the ISAs during their lab hours,
• join the instructors for Q&A sessions,
• talk to an instructor during their office hours, or
• write us an email

Piazza etiquette

• read the official assignment post before asking a question
• search to see if your question has already been asked
• use meaningful titles
• ask clarification questions for assignments (do not ask leading questions for Gold questions)
• do not discuss strategies for Gold questions
• do not post any of your assignment code publicly
• you can post your commented code privately, and an ISA or Instructor may provide some assistance
Course Learning Goals

At the end of each Section there are learning goals for the Section (in this Section, we present the learning goals for the entire course).

These learning goals clearly state what our expectations are.

Not all learning goals can be achieved just by listening to the lecture. Some goals require reading the text or using Seashell to complete the assignments.

Course Learning Goals

At the end of this course, you should be able to:

- produce well-designed, properly formatted, documented and tested programs of a moderate size (200 lines) that can use basic I/O
- use imperative paradigms (e.g., mutation, iteration) effectively
- explain and demonstrate the use of the C memory model, including the explicit allocation and deallocation of memory
- explain and demonstrate the principles of modularization and abstraction
- implement, use and compare elementary data structures (structures, arrays, lists and trees) and abstract data type collections (stacks, queues, sequences, sets, dictionaries)
- analyze the efficiency of an algorithm implementation