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.

Instructors

Your Winter 2021 instructors are:

Tim Brecht, Urs Hengartner, Dan Holtby, Gang Lu, Mark Petrick, Adrian Reetz, and Dave Tompkins.

It does not matter which section you are in or which instructor has been assigned to your section.

All instructors are sharing responsibilities for all students.

Conversely, you have access to all course instructors. For example, you may attend any office hour, regardless of your assigned instructor.

Additional course personnel:

- ISC (Instructional Support Coordinator)
- ISAs (Instructional Support Assistants)
- IAs (Instructional Apprentices)
- TAs (Teaching Assistants / Markers)

Course website

The course website:

`https://student.cs.uwaterloo.ca/~cs136/`

Has lots of useful links and resources.

The ***How Do I...?*** web page is a good starting point and explains how do accomplish some course-related tasks and find additional resources.

There is also a ***request system*** for submitting requests.

Announcements

All course announcements will be made on piazza.

> Pinned piazza posts are *mandatory* reading.

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	$\Rightarrow$	CS 136	$\Rightarrow$	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.

CS 136: Remote Learning

	Traditional	**Remote Learning**
Content Delivery:	lectures	course notes (readings)
(optional) Alternatives		voiceover videos synchronous lectures
(optional) Enrichment	tutorials	Q&A sessions supplemental videos
Self Assessment:	clickers	self-check quizzes
Applied Learning:	assignments	assignments
Formal Assessment:	exams	exams

Course Notes

These course notes will be the **primary source of information** and are available free online (at the course website).

> It is **your responsibility to thoroughly read these course notes**.

If you prefer, you may watch the *optional* voiceover videos or attend a live ("synchronous") lecture.

There are also *optional* supplemental videos to explain some topics in more detail.

Several different styles of “boxes” are used in the course notes:

Important information appears in a thick box.

Comments and “asides” appear in a thinner box. Content that only appears in these “asides” will **not appear on exams**.

Additional **“advanced”** material appears in a “dashed” box.

The advanced material enhances your learning and may be discussed in class and appear on assignments, but you are **not responsible for this material on exams** unless your instructor explicitly states otherwise.

Optional textbook

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

The textbook is **not** required, and is only recommended if:

- you want a different perspective on some of the material
- you desire additional examples
- you wish to learn more about the C language including topics not covered in this course (for use *after* this course)

Q&A Sessions

- Question & Answer (Q&A) sessions are hosted by instructors to answer questions related to the content in these course notes
- Q&As are open to all students and can be attended by multiple (100+) students at a time
- Q&A sessions are held on Microsoft Teams
- Read through these course notes *before* participating in a Q&A

Q&As are *not* for assignment-specific questions, which can be asked via piazza or during individual (one-on-one) office hours.

Office Hours

- individual (1-on-1) meetings with Instructors, ISAs and IAs
- all types of questions (*e.g.,* assignment and/or course content)
- either *general* (next available staff member will help you) or *specific* (scheduled office hours with an individual staff member)
- held on Microsoft Teams: enter the appropriate *queue* and someone will give you a “thumbs up” and call you when it is your turn (first come, first served)
- be prepared to share your screen

Schedule

Check piazza each week for the current schedule.

The *proposed* schedule is: (subject to change)

- **Friday:** new content available

  (course notes, self-check quiz, assignment)

- **Monday:** (optional) synchronous lecture

- **Wednesday:** self-check quiz due

- **Thursday:** assignment due

Q&A sessions and office hours occur throughout the week.

Marking scheme

- 50% assignments (weekly, weighted equally)
- 5% assignment style marks
- 5% self-check quizzes (weekly, weighted equally)
- 15% midterm exam
- 25% final exam

If your assignment average is not at least 50% you can not pass this course: your maximum achievable overall grade will be 46%.

Your assignment average does not include style marks.

Assignments

Assignments are *weekly* and each assignment is weighted equally (except A0).

- **read the assignment instructions carefully**
- read the assignment clarifications & FAQ
- read the assignment rules
  (new rules are added as the course progresses)

A0 does not count toward your grade, **but must be completed** before you can receive any other assignment marks.

Assignment Collaboration

Assignment questions are individually colour-coded as either **Black** or **Gold** to indicate the level of collaboration permitted.

For **Black** questions, **moderate collaboration** is permitted.

For **Gold** questions, **no collaboration** is permitted.

Test cases and documentation are part of your assignment and follow the same collaboration rules.

Black Assignment Questions

For **Black** questions (moderate collaboration):

- you may discuss assignment *strategies* openly
- you may search the Internet for strategies or code examples
- you may discuss or show your code with an *individual*, but **not** with a larger group (piazza, facebook, chat rooms, online forums, *etc.*)
- You may show your code to other individuals to give or receive help, but **copying is never allowed** (electronic transfer, copying code from the screen, printouts, *etc.*)

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**.

Piazza posts regarding Gold questions must be private (*post to: Instructors*).

Integrity

If you submit any work that is not your completely your own (*e.g.,* you receive help) you must cite (identify) the source of the assistance in an integrity statement. Even though moderate collaboration is allowed on **Black** questions, you must still cite any assistance or collaboration.

You do *not* have to cite any assistance you receive from course staff (including office hours, piazza posts, videos, course notes, *etc.*).

If you do not cite a source of assistance, you are presenting work of others as your own; this is called *plagiarism* and constitutes a violation of academic integrity (policy 71).

Assignments: second chances

Assignment deadlines are strict, but for some assignment questions you may be granted a "second chance" to re-submit your code.

- some assignment questions will have an automatic (or guaranteed) second chance
- other assignment questions may be granted a second chance based on the quantity and quality of submissions
- second chances are due 48 hours after the original deadline (unless posted otherwise)
- 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 supplemental videos for how to use `Seashell`.

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* (*e.g.,* for style), 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.

Assignment style marks

Assignment style marks are evaluated based on your adherence to the **course style guide** (available on the cs136 website).

The first four assignments (A1–A4) will not be marked for style, but you will still receive feedback.

Of the remaining six assignments, some assignments will be marked for style.

More details will be made available on piazza closer to A5.

View your style feedback on `MarkUs`.

You are not marked for style to “punish” or “torture” you. It is formative feedback to improve both your learning and the readability of your code.

You should follow the style guide even if your work is not being marked for style.

If your code has bad style, course staff may not provide assistance during office hours or on piazza.

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).

Self-check quizzes

There will be weekly quizzes to encourage active learning and provide timely feedback that allows you to assess your understanding of the course content.

Self-check quizzes are posted on LEARN and cover the weekly content from the course notes.

Self-check quizzes are only marked for *participation*, not correctness.

If you complete every self-check quiz on time, you will receive perfect self-check marks.

Support – Course Content

If you are struggling with the course content (concepts and material from the course notes) you may:

1. participate in a Q&A
2. ask a question during a synchronous lecture
3. post a public question on piazza
4. ask a question during individual office hours

Support – Assignments

If you are struggling with an assignment you should:

1. carefully re-read the assignment
2. read the assignment clarifications and FAQ
3. re-read the assignment and the FAQ one more time
4. **search** piazza to see if your question has already been asked
5. post a private question on piazza
6. get assistance during individual office hours

Support – Assignments (Black)

For **Black** questions, you may additionally:

- get help from a fellow student
- search the Internet
- post a public question on piazza (but do not post your code)

Support – Other

If you need help with something that is not related to the course content or an assignment (*e.g.,* an administrative issue):

- file a request (via the website)
- post on piazza
- discuss the issue during office hours
- write an email to the course ISC

Piazza etiquette

- **read** assignments & FAQs **thoroughly** before asking a question
- **search** to see if your question has already been asked
- **use** meaningful titles (*not* just “A1Q1”)
- **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 code (with **good style**) *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 reading the notes. Some goals require completing 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