Welcome

CS135 Lecture 00

Introduction to Course Personnel

Instructors: Zahra Ahmed

ISC (Instructional Support Coordinator): Deven Wolff (and Karen Anderson)

- The ISC handles course administration, including student illnesses and absences

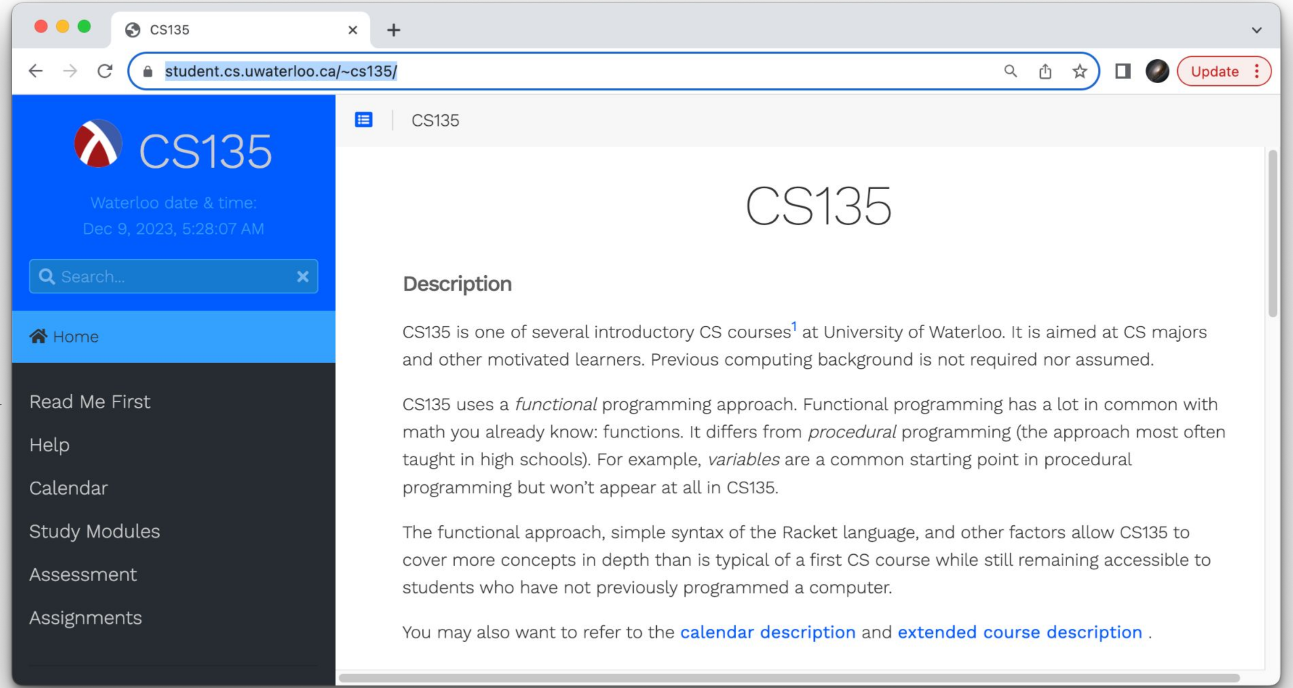
Other course personnel (see website for details):

- IAs (Instructional Apprentice)
- ISA (Instructional Support Assistants)
- TAs (Teaching Assistants)

Web site (main information source): `https://student.cs.uwaterloo.ca/~cs135`

https://student.cs.uwaterloo.ca/~cs135

Read me first



CS135

Description

CS135 is one of several introductory CS courses[1] at University of Waterloo. It is aimed at CS majors and other motivated learners. Previous computing background is not required nor assumed.

CS135 uses a *functional* programming approach. Functional programming has a lot in common with math you already know: functions. It differs from *procedural* programming (the approach most often taught in high schools). For example, *variables* are a common starting point in procedural programming but won't appear at all in CS135.

The functional approach, simple syntax of the Racket language, and other factors allow CS135 to cover more concepts in depth than is typical of a first CS course while still remaining accessible to students who have not previously programmed a computer.

You may also want to refer to the calendar description and extended course description .

CS135: Designing Functional Programs

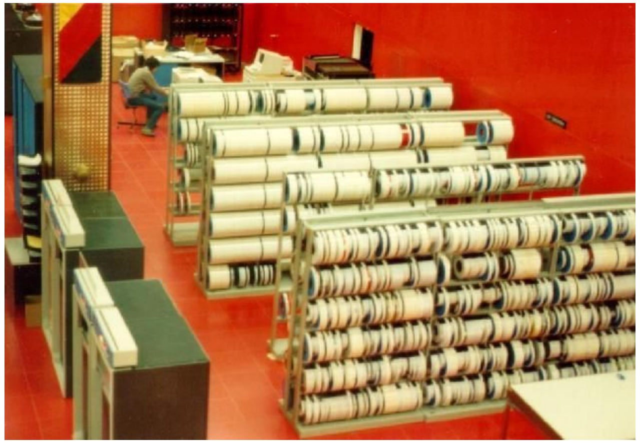
Your phone, laptop, desktop, etc. all contain one or more computers.

In this first Computer Science course, we don't just study computers or computer programming, we study computation itself. We take the first steps in an exploration of the fundamental properties and limitations about what can be computed, how it is computed, and how fast it can be computed. This exploration will be continued in later Computer Science courses.

The earliest computers, the computer in your watch, and the fastest gaming GPU all share the same fundamental properties and limitations.

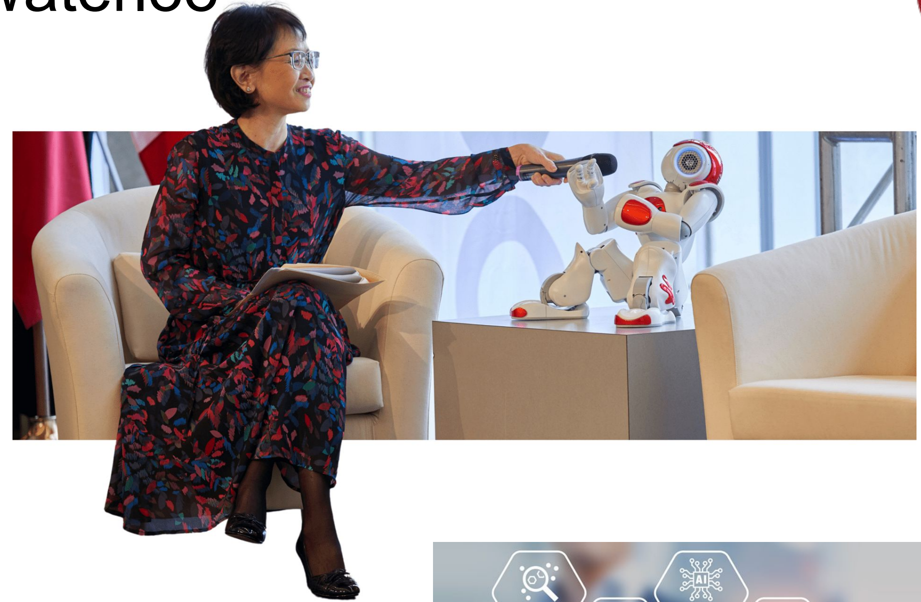
To study computation, we will use a simple, mathematically oriented functional programming model of computation, whose roots predate modern computers.

The "Red Room" in the MC Building, 1967-1999



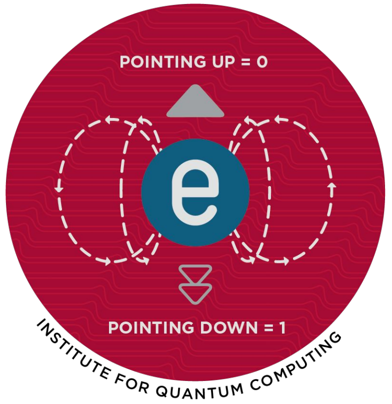
https://ist.uwaterloo.ca/cs/redroom

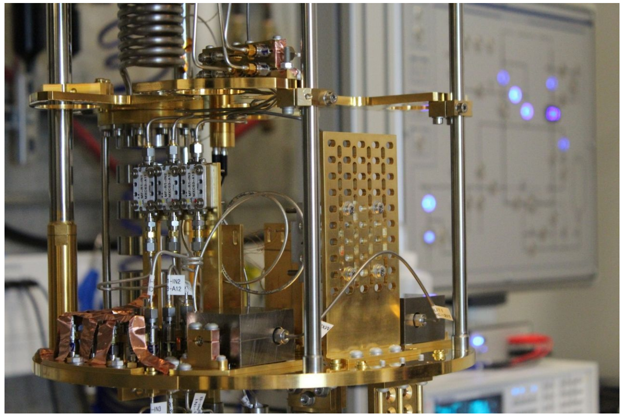
Artificial Intelligence at UWaterloo



https://uwaterloo.ca/artificial-intelligence-institute

Quantum computing at UWaterloo





https://uwaterloo.ca/institute-for-quantum-computing

Course Delivery

Lectures: Instructors present the core course material. Slides are available on the web site along with other supporting material. Lectures will include "clicker questions" to help you assess your understanding. Participation marks from these clicker questions form part of your course grade.

Tutorials: Course staff will work through problems that are similar to core material on the upcoming assignment. They will "think aloud" to show their thought processes and demonstrate how to apply our design recipe to help solve these problems.

Office Hours: Instructors and other course staff hold regular office hours, both online and offline, in which you can receive more individualized help. You may attend any office hour held by any instructor or staff member (not just your own instructor).

Piazza: A discussion forum for questions and messages to the class.

Assignments: Individual work to demonstrate what you’ve learned and receive feedback.

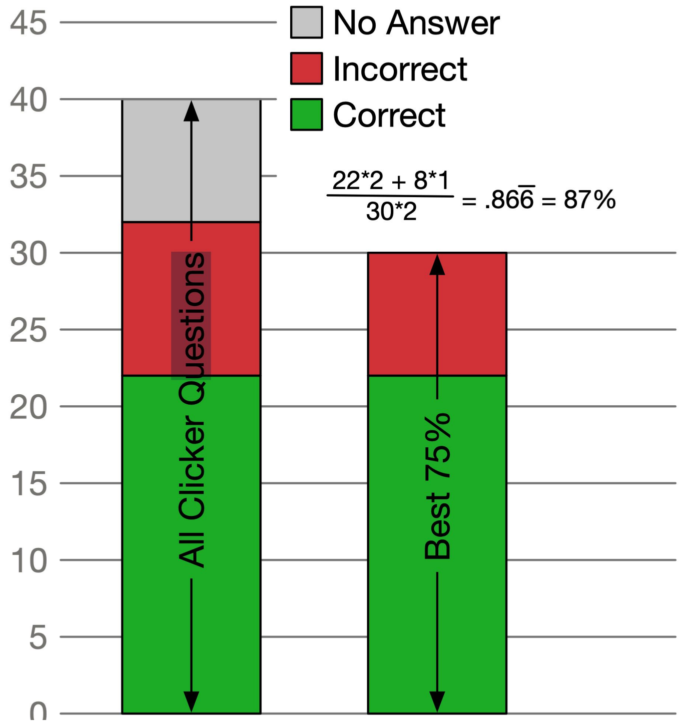
Exams: An opportunity for us to assess what you’ve learned.

Participation marks from clicker questions

Clicker questions encourage active learning and give you immediate feedback on your understanding.You respond to clicker questions using iClicker Cloud web or phone app. After all responses are collected, we look at the results and discuss.

Register at https://student.iclicker.com.

Marking: two marks for the correct answer; one mark for any other answer. We take the best 75% across the entire term to calculate 10% of your final mark. We take the best 75% to account for co-op interviews, sick days, etc. **You must attend the lecture you are registered in to earn the marks.**



$$\frac{22*2 + 8*1}{30*2} = .8\overline{66} = 87\%$$

Tutorials

- Held on Fridays in smaller groups than lectures. The first tutorial (this Friday) will be informal, providing help on A00.
- Led by an IA (Instructional Apprentice) or ISA (Instructional Support Assistant).
- Work through one or two complete examples in more detail.
- We "think aloud" while solving problems to show the complete process.
- Problems are not published. Often similar to the upcoming assignment problems. Lots of opportunities for questions.
- Bring your laptop.
- **You should definitely attend if your assignment marks are less than 80%.**

Assignments

Timing: 10 assignments, typically due Tuesday at 9:00pm.

Software: DrRacket (`https://racket-lang.org`)

Computer labs: MC 2062, 2063, 3005, 3027. Available for your use, but not scheduled labs. Most students use their own computers.

A00: Due soon. Must complete before the due date for Assignment 01 to receive marks for Assignment 01.

Submission: To the MarkUs system. More in A00. Submit early; submit often.

Email submissions will not be accepted. (Really. We mean it.)

We write programs in a language called Racket

Racket is a relatively recent dialect of a language called **Scheme** invented in the 1970s. Scheme is itself a cleaned-up version of **Lisp**, a language created in the late 1950s based on mathematical ideas (**λ calculus**) from the 1920s and 1930s.

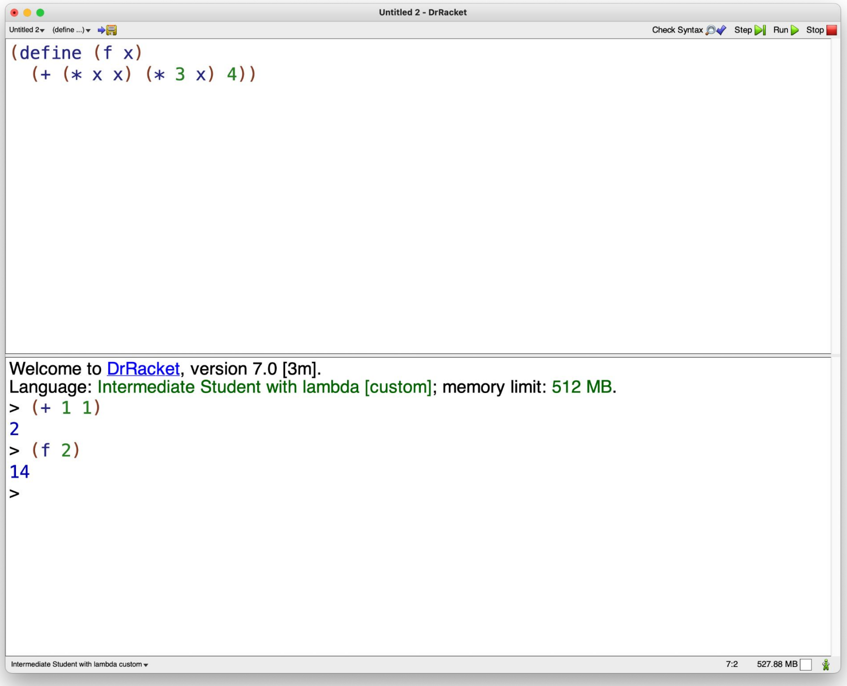
Racket is elegant and simple: We can describe the way it works completely and rigorously. It's also what's called a **pure functional language**, which is a more math-like way to program than other languages.

Racket lets you experiment with programming without getting bogged down in complicated syntax. In languages like C++ it's hard to write a legal program, never mind one that does what you want. We prefer to get to the meat of computer science faster.

We use a programming environment called DrRacket

- Provides a sequence of language levels
- Two panes:
  - Definitions (top) used for writing programs
  - Interactions (bottom) used for testing, experimenting

Under the language settings,select Choose Language…



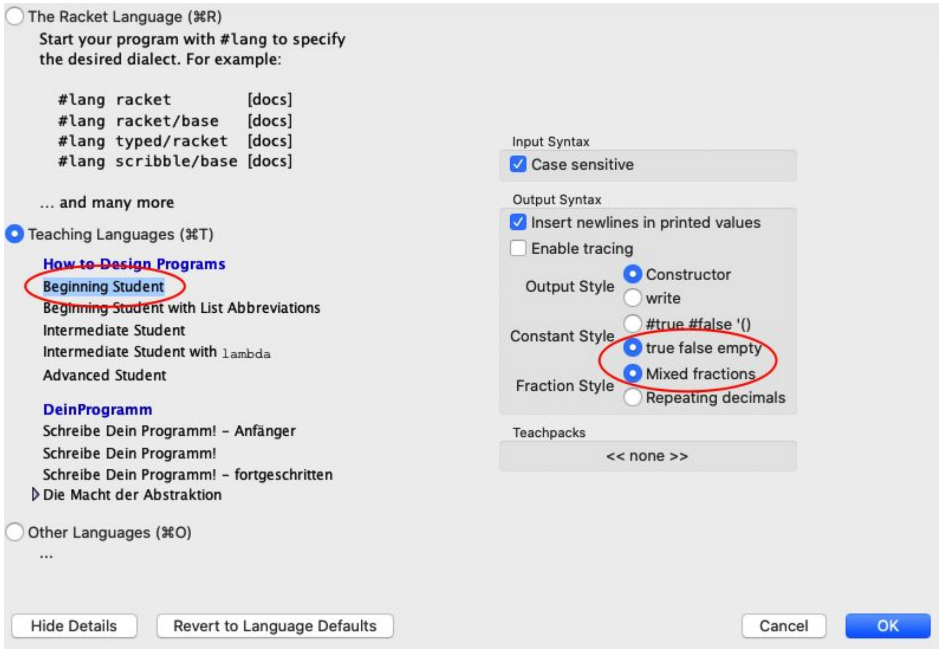
```
(define (f x)
  (+ (* x x) (* 3 x) 4))
```

```
Welcome to DrRacket, version 7.0 [3m].
Language: Intermediate Student with lambda [custom]; memory limit: 512 MB.
> (+ 1 1)
2
> (f 2)
14
>
```

Language settings in DrRacket

CS135 will progress through the Teaching Languages starting with Beginning Student. Follow these steps each time we change the language:

1. Under the Language settings, select Choose Language…
2. Select Beginning Student under Teaching Languages
3. Click the Show Details button in the bottom left
4. Under Constant Style, select “true false empty”
5. Under Fraction Style, select “Mixed fractions”



○ The Racket Language (⌘R)
Start your program with #lang to specify the desired dialect. For example:

```
#lang racket          [docs]
#lang racket/base     [docs]
#lang typed/racket    [docs]
#lang scribble/base   [docs]
```

… and many more

● Teaching Languages (⌘T)

How to Design Programs
[✓ Beginning Student]
Beginning Student with List Abbreviations
Intermediate Student
Intermediate Student with `lambda`
Advanced Student

DeinProgramm
Schreibe Dein Programm! – Anfänger
Schreibe Dein Programm!
Schreibe Dein Programm! – fortgeschritten
Die Macht der Abstraktion

○ Other Languages (⌘O)
…

Input Syntax
☑ Case sensitive

Output Syntax
☑ Insert newlines in printed values
☐ Enable tracing

Output Style: [✓ Constructor] write

Constant Style: #true #false '() [✓ true false empty]

Fraction Style: [✓ Mixed fractions] Repeating decimals

Teachpacks
<< none >>

Hide Details | Revert to Language Defaults | Cancel | OK

Assignments are individual work

You can talk to other students about:

- General course concepts
- Examples from class or tutorial
- Assignments problems 72 hours **after** the due date (to allow for any late submissions)

You may not share with other students any material you have written for current assignment problems, nor read any material they have written. You may not seek answers to specific problems on the Web, through any homework help service, or using AI tools, such as ChatGPT. In later assignments we may provide material you can use, but otherwise assignment solutions must be entirely your own work.

If you need help, we hold regular and frequent office hours.

Isn’t computer programming done by AIs now?

These days, most software developers employ *AI assistance* for software development. It’s basically ubiquitous.

Using AI assistance it’s possible to “vibe code” simple applications and web sites, generating software without a clear (or possibly any) understanding of how it works.

However, creating more complex software requires a fundamental understanding of how things work and how thing connect with real-world requirements.

While we are actively working to incorporate AI assistance into our upper-year courses, this course focuses on the fundamentals.

Exams

Midterm exam: March 2 at 19:00, 110 minutes in length

Final exam: Date to be determined by the Registrar. 2.5 hours in length.

The course was re-structured in 2025 and old exams (including exams from 2024) may no longer fully reflect the current version. We will make study material available closer to the exam dates.

Marking scheme

Participation marks: 10%

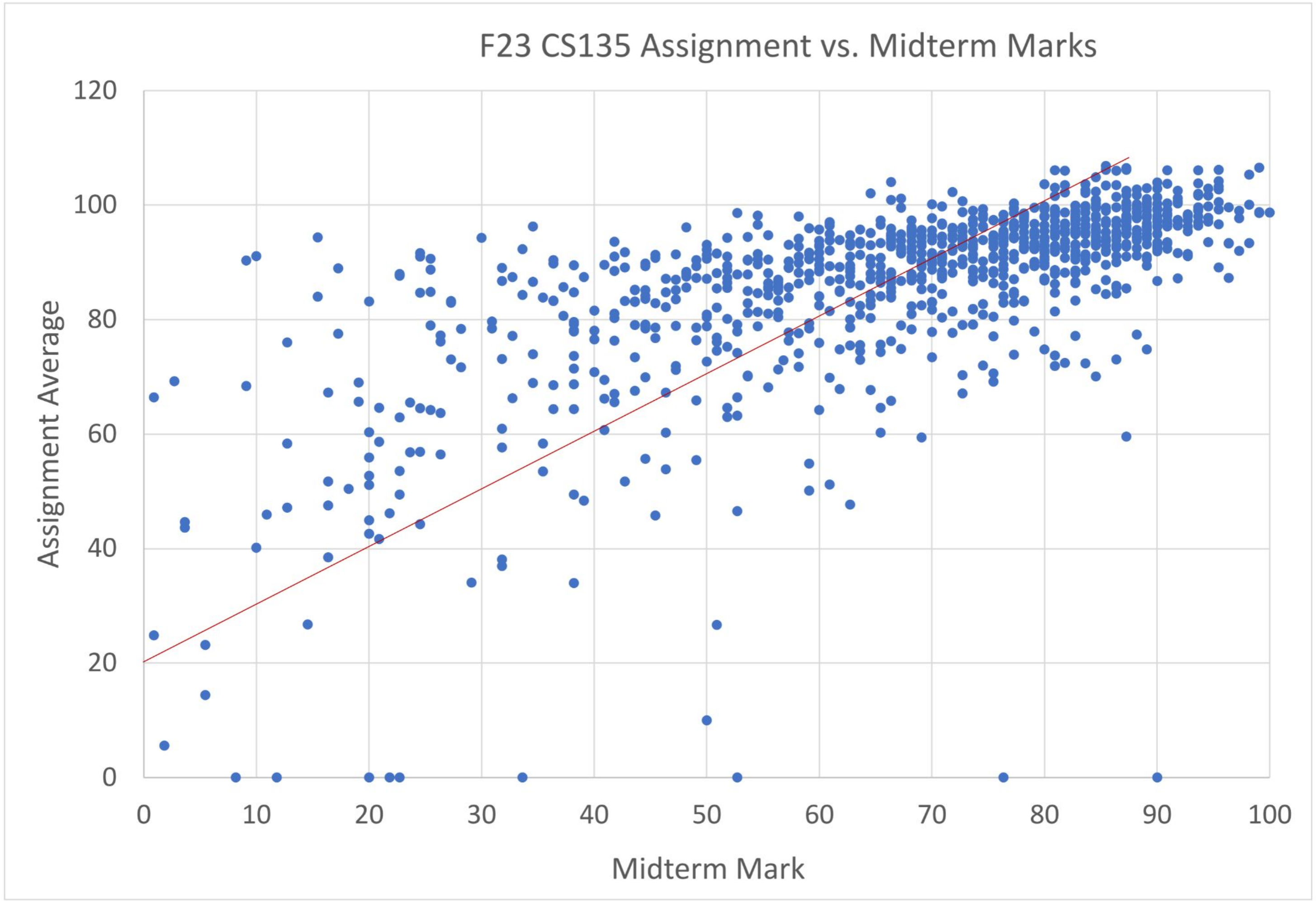
Assignments: 20%

Midterm exam: 25%

Final exam: 45%

To pass the course you must:

- earn at least half the available assignment points (10% out of 20%), **and**
- earn at least half of the available exam points (35% out of 70%).


F23 CS135 Assignment vs. Midterm Marks
Assignment Average
Midterm Mark
0
20
40
60
80
100
120
0
10
20
30
40
50
60
70
80
90
100

Our assignment and exam philosophy

Assignments and exam problems are structured so that it should be possible to obtain a grade of 60% on the course (enough to move forward to CS136) by demonstrating a solid understanding of core concepts.

On the other hand, it's truly challenging to get above 90% on the course.

As much as 50% of the marks on any assignment are "exercises", similar to the examples shown in class and tutorial, which focus on core concepts. Later assignments also contain "project questions" which build on the core material and challenge your to think on your own. Project questions may build on each other across the term and may also have bonus components and extensions, which encourage independent learning for interested people.

CS 135 vs. CS 115 vs. CS 145

CS 115 covers approximately two-thirds of the material in CS 135 at a slower pace. Students completing CS 115 then take CS 116 before moving on to CS 136, if they wish.

1A students in CS 135 are permitted to switch to CS 115, with no penalty, up to the deadline of October 22 during a Fall term, or February 14 during a Winter term.

CS145 is usually only offered in Fall terms. CS145 covers all the material in CS 135, plus additional material for students who want a deeper understanding. If you are interested, and there's availability, you can ask to be added to the CS 145 waiting list.

Communicating with us

- Ask questions in class – many questions are relevant to the whole class.
- Talk to your instructor after class.
- Instructors and ISAs hold office hours throughout the week.
  - In person and online via MS Teams
  - Times posted at: CS135 > Help > Office and Consulting Hours
  - Instructions for MS Teams at: CS135 > Help > Using Microsoft Teams
- We use Piazza for online discussion and Q&A:
  - Use meaningful subjects (not just “A3 problem"; what’s your specific problem?).
  - Search previous posts before posting; Don’t duplicate!.
  - Possible to post privately if necessary.
- Read your mail sent to your UW email account. We will only send to and reply to your UW email account. We do not respond to non-UW email addresses.

Copyright and intellectual property

The teaching material used in CS135 is the property of its creators, including:

- These course slides
- Assignment questions and solutions
- Exam papers and solutions

Do not share or post course material, especially assignment and exam material, on external websites, social media, chat groups and similar places.

Suggestions for success

- Read the CS135 ~~Survival~~Thrival Guide as soon as possible.
- Keep up with your assignments. Start them early. **This is key!**
- Go over your assignments and assessments; learn from your mistakes.
- Visit office hours as needed; earlier is better.
- Follow our advice on approaches to writing programs.
- Integrate exam study into your weekly routine.
- Maintain a “big picture” perspective: look beyond the immediate task or topic.

“Big picture” overview of the course

The course is structured into two halves, before and after the midterm:

The first half focuses on fundamentals:

- Our model of computation: values, substitution, functions, conditionals
- Recursion on natural numbers and lists

The second half extends these fundamentals:

- Structuring data with lists, especially lists of lists
- Higher order functions, including **`lambda`**

It’s important to have a solid foundation in place before starting the second half.

What happens next?

Over four lectures we will develop our model of computation:

1. Values and expressions
2. Functions
3. Conditional expressions
4. Recursion

After the final step, we will have built a complete “computer”, essentially from math.

We will then add “lists” to our model of computation to simplify data organization.

We will then explore a variety of basic algorithms and data structures using lists.

What to do next?

- Register your iClicker (https://student.iclicker.com)
- Download DrRacket (https://racket-lang.org)
  - Set language level
  - Try `(+ 1 1)` in interactions pane
  - Try `(+ 1 1)` in the definitions pane (don't forget to hit "Run" in the top right)
  - Try saving and restoring your work in the definitions panel
- Complete Assignment #0 (A00)