Announcements

- The times and locations of office hours are posted on the “Office & Consulting Hours” page of the course website. Please email us at cs135@uwaterloo.ca to set up an appointment outside of these hours.
- Assignment 1 is due on Tuesday, September 18, at 9:00 pm. Submit early and often to MarkUs! Check your basic test results after each submission. We will not be lenient about any late submissions.
- Make sure you complete Assignment 0 before the Assignment 1 deadline, if you haven’t done so already!
- Ensure that your clicker marks posted on the “View Marks” page of the course website are accurate.

Announcements

- MarkUs Basic tests:
  - Are set up for every assignment.
  - Do not thoroughly test your code.
  - Ensure we can run more thorough tests on your code after the due date.
  - The results are automatically emailed to your UWaterloo email. You can also check the results on MarkUs after each submission.
  - Are not related to the tests that you will write in your solutions.
Goals of this tutorial
You should be able to...

• Trace through the steps of evaluating simple Racket arithmetic expressions involving functions and constants
• Give a direct translation of mathematical expressions and functions in Racket.
• Understand when and how to use constants.
• Write the full design recipe for simple arithmetic functions.

How to find the help pages
• Do not use Google search. It will land you at the wrong language level, which is typically the full Racket help page.
• Open DrRacket: Help menu > Help Desk or Racket Documentation (this opens a browser window) > Teaching > How to Design Programs Languages > Select the appropriate language level (e.g. Beginning Student).
• Note the categorized list of functions on the left side bar.
• If you must use Google, then add the teaching language name to your query, e.g. “Racket beginning student”.

Clicker Question: Errors in Racket
Which of the following is an error-free Racket expression in “Beginning Student”?

A (8 + 6 / 3)
B (* (+(− 6 − 12 18)) − 24)
C (/ 5 (− 4 (sqrt 16)))
D (* (+ 5 10) (15))
E They all have some kind of error.
Stepping Problem
The following definition has been processed in the Beginning Student language:
(define y 1)
(define (g x)
  (+ x (* x x)))
Step through the following:
(g (g y))

Direct Translations: Tips
When given a mathematical function or expression to translate:

- Do not swap the order of parameters, or change their names.
  - For example, $2x + 2y$ should not be translated as
    $$(+ (* 2 y) (* 2 x))$$ or $$(+ (* 2 a) (* 2 b))$$.

- A mathematically equivalent expression is not necessarily a direct translation.
  - For example, $\frac{2}{x}$ should be translated as $\frac{1}{x} y$, but not
    $$(* x (expt y -1))$$.
  - Similarly, $2x + 2y + x$ should be translated as
    $$(+ (* 2 x) (* 2 y) x)$$, but not $$(+ (* 3 x) (* 2 y))$$.

Clicker Question: Direct Translation
What is the correct Racket translation of the following mathematical expression?
$$15 + \frac{(6+3)^2}{10} - 18 \cdot 17$$

A $$(+ (- 15 (sqr (/ (+ 6 3) 10))) (* 18 17))$$
B $$(+ (- 15 (sqr (/ (+ 6 3) 10))) (* 18 17))$$
C $$(+ (- 15 (/ (sqr (+ 6 3)) 10)) (* 18 17))$$
D $$(+ (+ 15 (/ (sqr (+ 6 3)) 10)) (* 18 17))$$
E None of the above
Group Problem: Direct Translation

Translate the following mathematical function into Racket:

The area of a regular polygon, given the length of one side, \(s\), and the number of sides, \(n\), can be computed with the following formula:

\[
polygon-area(s, n) = \frac{1}{4} \cdot n \cdot s^2 \cdot \frac{1}{\tan\left(\frac{\pi}{n}\right)}
\]

(Hint: \(\tan\) is a built-in Racket function that may be useful.)

Ensure that you are giving a direct translation of the function above.

---

Review: Advantages of constants

- Can give meaningful names to useful values (e.g. interest-rate, passing-grade, and starting-salary).
- Reduces typing and errors when such values need to be changed.
- Makes programs easier to understand.
- Constants can be used in any expression, including the body of function definitions.
- Sometimes called variables, but their values cannot be changed (until CS 136).

---

The Design Recipe on Assignments

Although the design recipe is not required for Assignment 1, all subsequent assignments will require you to write the design recipe.

A significant portion of your assignment marks will be from the design recipe!
Review: The five design recipe components

**Purpose:** Describes what the function produces. You should include parameter names in your purpose statement in a meaningfully way.

**Contract:** Describes what type of arguments the function consumes and what type of value it produces.

**Additional contract requirements:** If there are important constraints on the parameters that are not fully described in the contract, add an additional requires section to "extend" the contract.

**Examples:** Illustrate the use of the function.

**Definition:** The Racket definition (header and body) of the function.

**Tests:** A thorough set of function arguments and expected function values.

---

**Clicker Question - Design Recipe**

For the assignment function `final-cs135-grade`, which of the following has the best purpose, contract and example?

A: ```racket
;; (final-cs135-grade mid1 mid2 final assn) Calculates the cs135 final grade as a percentage using mid1, mid2, final, and assn
;; final-cs135-grade: Int Int Int Int -> Int
;; Example:
(check-expect (final-cs135-grade 100 100 100 100) 100)
```

B: ```racket
;; (final-cs135-grade mid1 mid2 final assn) Calculates the cs135 final grade as a percentage using mid1, mid2, final, and assn
;; final-cs135-grade: Int Int Int Int -> Int
;; requires: 0 < = mid1 < = 100
;; 0 < = mid2 < = 100
;; 0 < = final < = 100
;; 0 < = assn < = 100
;; Example:
(check-expect (final-cs135-grade 100 100 100 100) 100)
```

C: ```racket
;; final-cs135-grade: nat nat -> nat
;; Purpose: Calculates the cs135 final grade as a percentage
;; check-expect (final-cs135-grade 100 100 100 100) 100
```

D: ```racket
;; (final-cs135-grade mid1 mid2 final assn) Calculates the cs135 final grade as a percentage using mid1, mid2, final, and assn
;; final-cs135-grade: Nat Nat Nat Nat -> Num
;; requires: mid1 < = 100
;; 0 < = mid2 < = 100
;; 0 < = final < = 100
;; 0 < = assn < = 100
;; Example:
(check-expect (final-cs135-grade 100 50 100 50) 70)
```

E: ```racket
;; (final-cs135-grade mid1 mid2 final assn) Calculates the cs135 final grade as a percentage using mid1, mid2, final, and assn
;; final-cs135-grade: Int Int Int Int -> Int
;; requires: 0 < = mid1 < = 100
;; 0 < = mid2 < = 100
;; 0 < = final < = 100
;; 0 < = assn < = 100
;; Example:
(check-expect (final-cs135-grade 100 100 100 100) 100)
```

---

**Purpose & Contract**

Use the help desk to create a purpose and contract for `even?`. 
Group Problem: evaluate-tutorial-leader

At the end of the term, CS135 students are asked to evaluate their tutorial leaders. For this question, there will be three criteria that students will base their evaluations on:

- How knowledgable their leader is about course content
- How prepared they were for delivering tutorials
- How likely their leader could win in a battle royal against the other tutorial leaders

Also, each criteria is weighted differently:

- Knowledge - 20%
- Preparedness - 5%
- Battle Royal - 75%

Students can rate each criteria on a scale of 1-10, where 1 is the lowest rating and 10 is the highest rating.

Group Problem: evaluate-tutorial-leader

Write a function `evaluate-tutorial-leader`, that consumes three parameters: A score from 1-10 based on the knowledge, preparedness, and battle royal criteria, respectively. The function produces the weighted score that the tutorial leader receives, according to the weightings on the previous slide. Include the full design recipe for this function.

```
(evaluate-tutorial-leader 8 7 1) => 2.7
(evaluate-tutorial-leader 5 1 10) => 8.55
```