Assignment 01
Due Wednesday, May 27 at 10:00 am (no late submissions)

!

Plagiarism.

- All work in CS 115 is to be done individually. The penalty for plagiarism on assignments (first offense) is a mark of 0 on the assignment and a 5% reduction of the final grade, consistent with School of Computer Science policy. In addition, a letter detailing the offense is sent to the Associate Dean of Undergraduate Studies, meaning that subsequent offenses will carry more severe penalties, up to **suspension** or **expulsion**.
- To avoid inadvertently incurring this penalty, you should discuss assignment issues with other students only in a very broad and high-level fashion. Do not take notes during such discussions, and avoid looking at anyone else's code, on screen or on paper. If you find yourself stuck, contact the ISA or instructor for help, instead of getting the solution from someone else. Do not consult other books, library materials, Internet sources, or solutions (yours or other students') from other courses or other terms.

Assignment Guidelines.

- This assignment covers material in Module 2.
- Submission details:
  - Solutions to these questions must be placed in files `a01q1.rkt`, `a01q2.rkt`, `a01q3.rkt`, and `a01q4.rkt`, respectively, and must be completed using Racket *Intermediate Student*.
  - All solutions must be submitted to MarkUs. No solutions will be accepted through email, even if you are having issues with MarkUs.
  - Verify using MarkUs and your basic test results that your files were properly submitted and are readable on MarkUs.
  - For full style marks, your program must follow the CS115 Style Guide.
  - Be sure to review the Academic Integrity policy on the Assignments page.
  - For the design recipe, helper functions only require a purpose, a contract and an example.
- When a function returns an inexact answer, use a tolerance of 0.0001 in your tests.
- Restrictions:
  - Unless the question specifically describes exceptions, you are restricted to using the functions and special forms covered in or before Module 2.
  - Read each question carefully for additional restrictions.
- **The solutions you submit must be entirely your own work. Do not look up either full or partial solutions on the Internet or in printed sources.**

Write your design recipe carefully, for every function you write.

- Consider any restrictions on the input.
- Use helper functions appropriately.

1. Design.

Exercise

Write the design recipe, **omitting the implementation**, for a function `(e-digits n)`.
It returns an approximation of $e \approx 2.71828$, correct to `n` digits.
Write examples, but comment them out with `;;`.

Exercise

Bernoulli numbers, which have many applications in Mathematics, are described here:
`https://en.wikipedia.org/wiki/Bernoulli_number`
Write the design recipe, **omitting the implementation**, for a function `(bernoulli n)`. It returns the `n`th Bernoulli number. For values that may be positive or negative, use the positive value.
Write examples, but comment them out with `;;`.

2. Areas.

Exercise

Write a function `(area-triangle b h)` that returns the area of a triangle with base `b` and height `h`.
For example,

```
(area-triangle 0.5 6) => 1.5
```

The "usual" equation for the are of a triangle is useful only if the base and height are known. Sometimes the three side lengths are known, but not the height. In this case, it is easier to use *Heron's Formula*:
Given a triangle with side lengths a, b, and c, the area of the triangle is

$$A_{\triangle} = \sqrt{s(s-a)(s-b)(s-c)}$$

where $s = \frac{a+b+c}{2}$ is the semi-perimeter of the triangle.

Exercise

Write a function `(area-heron a b c)` that returns the area of a triangle with side lengths `a`, `b`, and `c`.
For example,

```
(area-heron 4 13 15) => 24
(area-heron 2 2 2) => (sqrt 3) => #i1.7320508075688772
```

Note: for any real triangle, *the sum of any two sides is greater than the remaining one.* This is proved in Euclid's Elements, Book I, Proposition 20.
You cannot construct a triangle with side lengths 5, 2, and 2, for example.
Your function will give an imaginary result if you try:

```
(area-heron 5 2 2) => 0+3.75i
```

Imaginary numbers are not a part of this course, so work with real triangles. This would be a good place to add a `Requires`.

3. Modelling Fluid Flow. The Reynolds Number is a value used in the analysis of fluid flow in tubes. It is given by:

$$\text{Re} = \frac{uD}{\nu}$$

where u is the flow speed in metres per second, D is the diameter of the tube in metres, and ν is the kinematic viscosity of the fluid in square metres per second.

Write a function `(reynolds u D v)` that calculates the Reynolds number for the given parameters.

Carefully consider the parameters of the function. Do you need a `;; Requires` section, and if so, what does it need to contain?

4. Str. *Read the documentation in DrRacket on the functions* `min` *and* `max`*, and review the documentation on* `Str`*. Some of these functions will be required to complete this question.*

Write a function `(trunc8 s)` that consumes a `Str` and returns a `Str` of length at most 8, containing up to the first 8 characters of `s`.
A few examples:

```
(check-expect (trunc8 "Howdy") "Howdy")
(check-expect (trunc8 "Go ahead, make my day") "Go ahead")
```

Write a function `(truncfill s)` that truncates strings longer than 8, like `trunc8`, but also "pads" strings shorter than 8 by adding as many `"."` as necessary.
A few examples:

```
(check-expect (truncfill "Howdy") "Howdy...")
(check-expect (truncfill "Go ahead, make my day") "Go ahead")
```

We will not discuss `cond` until Module 4. Do not use it on this assignment. The necessary results can be achieved using some combination of `min`, `max`, and `Str` functions.