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 people’s) 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 a1q1.rkt, a1q2.rkt, a1q3.rkt, and a1q4.rkt, respectively, and must be completed using Racket.
  – 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.
  – Helper functions need design recipe elements but not examples and 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.
- Consider any restrictions on the input.
- Use helper functions appropriately.

1. **Quadratic Equation.** The roots of a quadratic equation of the form \( y = ax^2 + bx + c \) are given by

\[
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]

The portion \( b^2 - 4ac \) is sometimes called the *discriminant*.

**Exercise**
Write two functions: (root-plus a b c) and (root-minus a b c). Each function consumes three Num, and returns a Num. root-plus returns the more positive root of the equation, and root-minus returns the more negative root.
For example,
- (root-plus 1 -8 15) => 5
- (root-minus 1 -8 15) => 3
- (root-plus 3 -6 1) => #i1.81649658...
- (root-minus 3 -6 1) => #i0.183503419...

2. **Bernoulli Equation.** The Bernoulli equation describes the behaviour of incompressible fluids. In any given streamline,

\[
\frac{v^2}{2} + gz + \frac{p}{r} = \text{constant}
\]

Suppose for a given streamline that the constant is 12.3.

**Exercise**
Write a function (bernoulli-p v g z r) that consumes four Num, and returns the value of \( p \) for this streamline.
For example, if \( v = z = 0, r = 2, \) and \( g = 9.8, \)
- (bernoulli-p 0 9.8 0 2) => 24.6

3. **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.

**Exercise**
Write a function (pad3 n) that consumes a Nat and returns a Str. The Str contains the digits of n, with zeros added at the front to make it of length 3. Only the first three digits of numbers 1000 or greater are retained.
For example,
- (pad3 7) => "007"
- (pad3 42) => "042"
- (pad3 245) => "245"
- (pad3 3141592) => "314"

We will not discuss **cond** until Module 4. Do not use it on this assignment. The necessary effects can be achieved using some combination of **min**, **max**, and **Str** functions. The **Str** documentation is on the course website, listed under [general resources](#).