Basic Tests

- Write a function to convert degrees Fahrenheit to degrees Celsius.

\[ C(F) = \frac{5}{9} (F - 32) \]

- Note the error in the body of `fahr->celsius`. Which of the following `check-expects` will pass (if any)?

```scheme
(check-expect (fahr->celsius 32) 0) ; freezing point of water
(check-expect (fahr->celsius 212) 100) ; boiling point of water
(check-expect (fahr->celsius -40) -40) ; cross-over point
```

```scheme
(define (fahr->celsius degF)
  (* 5/8 (- degF 32)))
```
Basic Tests

- Likely Basic Tests for `fahr->celsius`:
  - `(check-expect (fahr->celsius 32) 0)
  - `(check-expect (number? (fahr->celsius 212)) true)`
Basic Tests

- Likely Basic Tests for fahr->celsius:
  - (check-expect (fahr->celsius 32) 0)
  - (check-expect (number? (fahr->celsius 212)) true)

Conclusions: Basic Tests are focused on

- Is your function named correctly?
- Does it consume the correct number and type of parameters?
- Does it produce an answer of the expected type?

Basic Tests are **not** particularly concerned with the correctness of your function. That is your responsibility.
Substitution Rules

Repeatedly rewrite the leftmost eligible subexpression with one of the following substitution rules until a value or error is obtained:

- \((f \ v_1 \ldots \ v_n) \Rightarrow v\) where \(f\) is a built-in function, \(v_1 \ldots v_n\) are values, and \(v\) is the value of \(f(v_1 \ldots v_n)\).
- \((f \ v_1 \ldots \ v_n) \Rightarrow \text{exp}'\) where (define \((f \ x_1 \ldots x_n) \ \text{exp}\)) occurs to the left, and \(\text{exp}'\) is obtained by substituting into the expression \(\text{exp}\), with all occurrences of the formal parameter \(x_i\) replaced by the value \(v_i\) (for \(i\) from 1 to \(n\)).
- \(id \Rightarrow \text{val}\) where (define \(id \ \text{val}\)) occurs to the left.

- \((\text{and} \ \text{false} \ldots) \Rightarrow \text{false}\)
- \((\text{and} \ \text{true} \ldots) \Rightarrow (\text{and} \ldots)\)
- \((\text{and}) \Rightarrow \text{true}\)
- \((\text{or} \ \text{true} \ldots) \Rightarrow \text{true}\)
- \((\text{or} \ \text{false} \ldots) \Rightarrow (\text{or} \ldots)\)
- \((\text{or}) \Rightarrow \text{false}\)

Where is the rule for \((\text{not} \ v)\)?
Rollercoaster Rules

- Riders must be at least 1.2 meters tall.
- Riders must be at least 12 years old or accompanied by an adult.
- Riders with a gold pass may ride, regardless of height or age.

Write a function, (able-to-ride? height age with-adult? pass), where
  - height is the rider’s height in meters (a number)
  - age is the rider’s age in years (a number)
  - with-adult? is true if the rider is accompanied by an adult and false otherwise
  - pass is one of 'gold, 'silver, or 'bronze

able-to-ride? produces true if the rider is allowed to ride and false otherwise.

We will solve this using three different approaches.
Rollercoasterer Summary

- We solved it three different ways:
  - A pure Boolean expression
  - With `cond`, focusing on conditions where the rider is able to ride
  - With `cond` and a more mixed or ad hoc set of conditions

- Learnings:
  - There may be many ways to solve a problem. Brainstorm them before you begin.
  - Having tests/examples available helps us get things correct.
  - Use constants for numbers like 12 and 1.2. Name them well.
  - `boolean=?` is rarely needed. It is usually banned in CS135.
  - Boolean identifiers like `with-adult?` can be used directly in Boolean expressions.
  - The order of the conditions in a `cond` matters.