CS135 Tutorial 02

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)?

```
(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

(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 v1 … vn) => v` where `f` is a built-in function, `v1 … vn` are values, and `v` is the value of *f(v1 … vn)*.
- `(f v1 … vn) => exp’` where `(define (f x1 … xn) exp)` occurs to the left, and `exp’` is obtained by substituting into the expression `exp`, with all occurrences of the formal parameter `xi` replaced by the value `vi` (for i from 1 to n).
- `id => val` where `(define id val)` occurs to the left.

- `(and false …)` => `false`
- `(and true …)` => `(and …)`
- `(and)` => `true`
- `(or true …)` => `true`
- `(or false …)` => `(or …)`
- `(or)` => `false`

Where is the rule for `(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.

Rollercoaster 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.