CS 135 Winter 2020

Tutorial 2: Boolean Logic and Conditional Expressions

Announcements

- The times and location of office hours are posted on the “Office and Consulting Hours” page of the course website. Please email us at cs135@uwaterloo.cato set up and appointment outside of these hours.
- Assignment 2 is due on **Tuesday, January 28, at 9:00 PM**.
- If you have not already done so, make sure to complete Assignment 0 **before** submitting any assignments!
- Drop deadline without penalty is **today**.

Goals of this tutorial

You should be able to...

- Understand the basics of **Boolean Algebra**.
- Implement **Conditional Statements** in Racket.
- Identify and correct errors in Boolean functions and Conditional Statements.
- Write the full design recipe for functions involving cond and the boolean operations.

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: Contracts

Which of the following options are a correct contract for the function cs135-grade-sofar that you wrote for Assignment 1?

A ;; cs135-grade-sofar: Num Num Num → Num

B ;; cs135-grade-sofar: Nat Nat Nat → Num

C ;; cs135-grade-sofar: num num num → num

D ;; cs135-grade-sofar: NAT NAT NAT → NUM

E ;; None of the above.

Review: Boolean-valued functions

Boolean-valued functions produce Boolean values: true and false.

These functions are also called **predicates**.

Standard Racket uses #t and #f, or #true and #false; these will sometimes show up in basic tests and correctness tests.

Racket provides many built-in Boolean functions (for example, to do numerical comparisons: (>= x y), (= x y)).

Review: Boolean Valued Functions

To determine if the proposition "a $<$ b" is true or false, we can write it as the Racket expression ($<$ a b).

There are also functions for $>$ $=$ $>=$ $<=$.

We can also combine multiple boolean functions using special forms and, or, not.

Example: "4 $<$ x $<$ 20" = (and ($<$ 4 x) ($<$ x 20))

Review: Boolean-valued functions

Note that comparison functions are often specific to certain data types (for example, (= a b) vs. (symbol=? x y), where a and b are numbers, but x and y are symbols).

The naming convention for most predicates and Boolean parameters is to append a question mark to the name (for example, even?, symbol?, expired?).

Review: Boolean Operators

and and or are special forms in Racket.

and and or may have two or more arguments.

Their arguments are evaluated from left to right.

and:

- If an argument evaluates to false, the entire expression evaluates to false.
- Otherwise, the next argument is evaluated.
- If there are no arguments remaining, the expression evaluates to true.

Review: Boolean Operators

or:

- If an argument evaluates to true, the entire expression evaluates to true.
- Otherwise, the next argument is evaluated.
- If there are no arguments remaining, the expression evaluates to false.

not:

- not must have exactly one argument.
- If the argument evaluates to true, the entire expression evaluates to false.
- If the argument evaluates to false, the expression evaluates to true.

Group Problem - receives-discount?

A warehouse store discounts its merchandise according to the following rules:

- If an item has been in the store for at least 6 weeks, it is only discounted if the item is an 'appliance or 'clothing.
- If an item has been in the store for at least 3 weeks, but less than 6 weeks, it is only discounted if the item is a 'food.
- All other items are not discounted.

Using only Boolean operations (and, or, or not), write a function receives-discount? that consumes the number of weeks an item has been in the store, and a symbol representing the type of the item. The function produces true if the item receives a discount, and false otherwise. You don't have to write tests for this function (yet).

receives-discount? - Tests

(check-expect (receives-discount? ?? ??) ??) What can we fill the blanks with?

Remember that our examples covered some possible cases (these are black box tests).

Examples:

```
(check-expect (recieves-discount? 5 'appliance) false)
(check-expect (receives-discount? 10 'food) false)
```

Review: Conditional Expressions

The general form of a conditional expression is:

```
(cond
   [question1 answer1]
   [question2 answer2]
   ...
   [questionk answerk])
```

where questionk could be else.

- Each of the questions must evaluate to a **boolean** value.
- The questions are evaluated from **top to bottom**.
- If a question evaluates to true, no more questions are evaluated and the cond expression is reduced to just the answer for that question.
- If none of the questions evaluate to true, then the result is the answer in the else clause.
- If there are no questions that evaluate to true and there is no else clause, then Racket will report an error.

Clicker Question - Conditional Errors

```
;; foo: Nat Sym → Nat
(define (foo bar baz)
    (cond
        [(false? baz) bar]
        [(= bar baz) (+ 1 baz)]
        [(+ 1 bar) bar]
        [(symbol=? baz 'meow) (/ bar bar) bar]))
```

How many errors are in the above function?

A 6
B 1
C 3
D 4
E This is Java code

Group Problem: FizzBuzz

FizzBuzz is a classic Computer Science problem often used in **interviews**.

Define a function that consumes a single number “n” and produces:

- “Fizz” if **n** is a multiple of 3
- “Buzz” if **n** is a multiple of 5
- “FizzBuzz” if **n** is a multiple of both 3 and 5
- The value of **n** if it is **not** a multiple of 3 or 5

Group Problem: Design Recipe

- Purpose:

  ;; (fizzbuzz n) Consumes a number n and produces the

  ;; proper response following the rules of ”FizzBuzz”.

- Contract:

  ;; fizzbuzz: Int → (anyof Str Int)

- Examples:

  (check-expect (fizzbuzz 3) "Fizz")

- Tests:

  (check-expect (fizzbuzz −5) "Buzz") (check-expect (fizzbuzz 17) 17)

  (check-expect (fizzbuzz 30) "FizzBuzz") (check-expect (fizzbuzz 0)

  "FizzBuzz")

Extra Practice - Design Recipe

We will not cover these questions in the tutorial, unless we have some time left over in the end. Solutions will not be posted to these questions - you may discuss these problems with classmates and work together to come up with a solution.

- Write the design recipe for the helper function divisible-by? that we wrote in the solution for the fizzbuzz question above. As a reminder, the function takes a base and a multiple and determines if the base perfectly divides the number.
- Write the design recipe for the functions you wrote in A01.

Extra Practice - Conditionals

We will not cover these questions in the tutorial, unless we have some time left over in the end. Solutions will not be posted to these questions - you may discuss these problems with classmates and work together to come up with a solution.

You've finally made it to the CS135 Gym and it is time to battle. Here are the rules:

- only 3 types are allowed: Fire, Water and Rock
- Water beats both Fire and Rock
- Rock beats Fire
- Any type can beat itself

Write a function pokemon-battle that consumes 2 Pokemon types and produces the winning type according to the rules above. Use the symbols 'fire, 'water and 'rock to represent the types. Include a design recipe.

Extra Practice - Boolean Functions (Part 1/2)

We will not cover these questions in the tutorial, unless we have some time left over in the end. Solutions will not be posted to these questions - you may discuss these problems with classmates and work together to come up with a solution.

The Iron Bank of Braavos has the following rules for setting a Bank PIN:

- A Bank PIN must be a 4-digit positive integer.
- The last two digits of a Bank PIN must be a multiple of 7, but cannot be divisible by 6 or 9.

For example, 1049 is considered a valid Bank PIN, but 9999 is not. **Using only boolean expressions for the entire question**, first write a helper function last-two-digits that consumes a number n and produces the last two digits of n.

Extra Practice - Boolean Functions (Part 2/2)

The Iron Bank of Braavos has the following rules for setting a Bank PIN:

- A Bank PIN must be a 4-digit positive integer.
- The last two digits of a Bank PIN must be a multiple of 7, but cannot be divisible by 6 or 9.

Next, write the function valid-pin? that consumes a number, and produces true if the number is considered a valid PIN according to the rules above, and false otherwise. Include a purpose, contract, and examples.