Tennis

A Racket Testing Suite

Theron Rabe. 2014.

Table of Contents

Tennis: A Racket Testing Suite

Introduction

Modern high-level programming languages often come packaged with unit-testing libraries. Python, Ruby, and D are examples of such languages. The Tennis testing suite focuses on another such language: Racket. Although a basic installation of the Racket framework contains unit-testing software (rackunit), the language does not contain any libraries that dynamically generate its test cases. The Tennis library aims to correct this limitation.

Why Racket? For verified testing methods. Since Racket is a pure functional language in which programs do not have mutable state, each function of a program may be implemented through execution of its formal specification. From a software engineering point of view, every syntactically valid substring of a program is guaranteed to be a mathematically-sound formal specification of itself, thus simplifying the transition from specification to implementation. The benefits of a pure functional testing suite also include simplification of formal verification methods such as module-boundary type contracting and use of automated theorem provers (ala Agda or Coq). These benefits of Racket help assure a Tennis user that their programs will be tested thoroughly and that those tests are completed in a provably correct environment. When you test with Tennis, you don't ever have to worry about state-related errors. This in combination with Racket's powerful macro system makes dynamic test generation require minimal code modification.

Tennis aims to be easy to integrate into any dialect of the Racket programming language, and straightforward to use with old projects without the need to modify much code. The Tennis system uses your code not only to generate functional test cases, but to generate entirely new Racket programs that serve to test as much or as little of your project as desired.

Installation

To install the Tennis collection in Racket, navigate a terminal to the directory containing the “tennis” subdirectory. Issue the command: “raco pkg install --link tennis”. The Tennis testing system is now available to your Racket programs.

To access your Tennis installation from your project, make a “tests” directory at the root of your project. In each module to be tested, add the line: “(require tennis)”. All of the functions provided by Tennis are now available for use in the module. When your program is executed, it can now automatically generate programs that test it. These tester programs will be located in the “tests” directory.

Beginner's Guide

Once Tennis is installed to Racket, it becomes available to your projects. Tennis is designed to require minimal code modification for integration with existing projects. To generate a Tennis tester for any Racket function should require the addition of only one or two lines of code in a module, and the alteration of none. Once Tennis has been integrated into a project, the first time execution of a module will generate a new Racket program that tests the specified functions against expected values. This tester is automatically integrated with the rackunit unit testing collection, to provide concise and verbose function test failures.

To integrate Tennis with a module, add the following line of code to the top of the module:

(require tennis)

Then, for each function that is to be tested, a domain needs to be defined, options need to be set for the output tester program, and the generateTester function needs to be called. A domain is defined in Tennis as any set of enumerable ranges, which in Racket terms is to say a domain is any expression of the following form:

(list (list a b _ …) …)

Each range in the domain correlates its associated argument to valid input values. The first and last elements in each range notate minimum and maximum, and intermediate elements represent suggested nominal values. An example domain for the inverse cosine function is thus: '((-1 1)). An example domain for a function that takes two arguments might be: '((1 10) (100 200)). After domains have been decided for the functions to be tested, Tennis can be invoked using the generateTester function.

The generateTester function provided by Tennis takes the form:

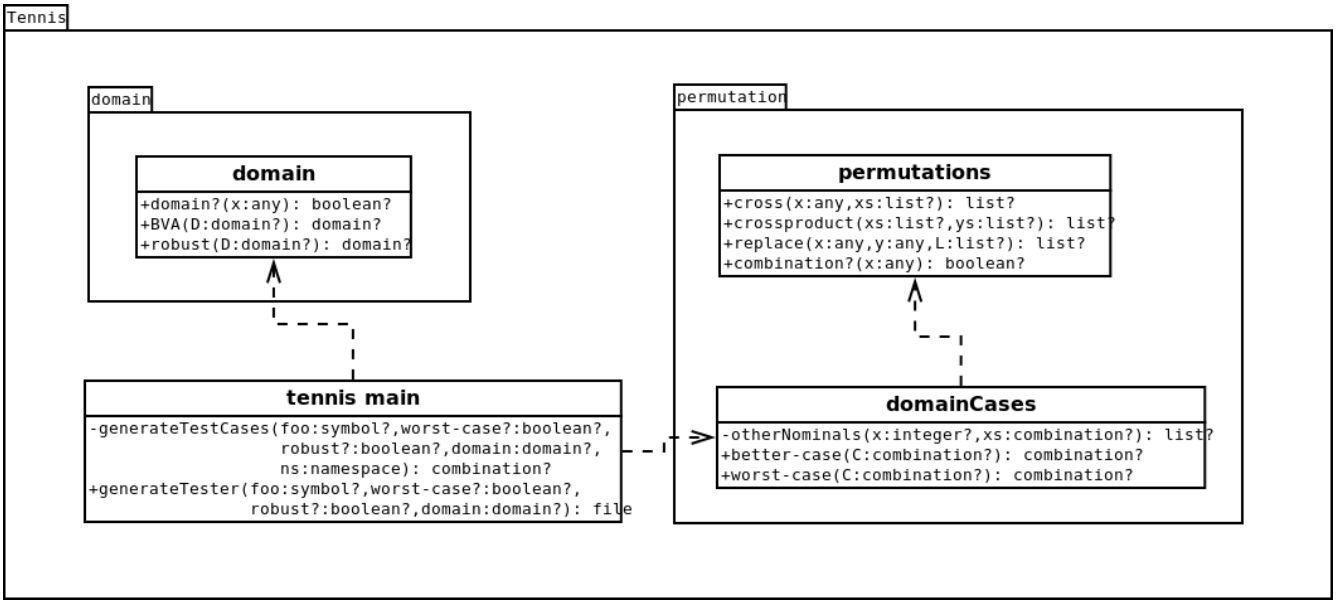
(generateTester 'functionName worst-case robust domain)

Where 'functionName is a quoted symbol that correlates to a function, worst-case is a boolean flag that indicates whether testing should be single-fault (#f) or worst-case (#t), robust is a boolean flag that indicates whether the test cases should cover a boundary-value analysis (#f) or a robust boundary-value analysis (#t), and domain is the domain over which the function is defined. When a call to this function is found in a module, Tennis will generate new Racket programs in the ./tests/ directory that test each of the indicated functions according to the passed flags. Each line of the output tester program will be a call to the rackunit unit-testing library to compare the actual output of the function to an expected output.

By default, the expected output as specified in a tester program is set equal to the actual output at the time of its generation. The expected outputs can easily be modified by replacing the last argument of each line. When a tester program is executed using Racket, it will verbosely output any encountered failures. In the upcoming documentation, the behavior of tester programs is exemplified.

Design Specification

Tennis is composed of a hierarchy of modules. Each module contains a collection of interrelated functions. Every function is pure, with no mutable state, and thus no side effects. A graphical representation of this hierarchy can be seen in this figure:



The main tennis module provides a function to generate tester programs, and relies on parts of the permutation and domain collections.

The domain collection provides the domain module, which defines the concept of a function domain, and provides functions to transform them. The “domain?” function returns true if its input is a domain, and false otherwise. The “BVA” function transforms a provided domain into a domain representing the input's boundary-value analysis. Similarly, the “robust” function transforms an input domain into a domain representing a robust version of its input.

The permutation collection provides two modules: permutations and domainCases. The permutations module provides mathematical set combinators “cross”, “crossproduct”, and “replace”. It also provides the definition of a combination. The domainCases module provides functions that return combinations of sets that satisfy the definition of a domain. These functions include “better-case” and “worst-case”. An additional function is defined for this module, but not provided externally: “otherNominals” which is an intermediate domain combinator used to calculate “better-case”.

Since Racket is purely functional, each function's formal specification is also an executable implementation. To understand Tennis' specification at a mathematical level of abstraction, continue reading to the implementation section of this document, which contains all functions' pre- and post-conditions, type contracts, and formal specifications, as expressed in Racket.

Implementation

To meet the specified modular hierarchy, Tennis is organized in a file system of directories that correlate to the main tennis, domain, and permutation collections. In each of these directories are Racket code files that correlate to each of the specified modules. For example, the functions provided by the domainCases module can be found in the file: tennis/permutation/domainCases.rkt.

At the top of each module is a list of provided functions, along with their type contracts. Each provided function is formally defined through pure functional specification later in the body of the module. The Racket source for each module is provided in the following pages.

Implementation of main.rkt

```
#lang racket

(require "domain/domain.rkt" "permutation/domainCases.rkt")
(provide (contract-out
        [generateTester (-> symbol? boolean? boolean? domain? any)]))

;
;generateTestCases returns a list of executable test calls according to a domain and the worst-case? and robust? flags.
;
    (define (generateTestCases foo worst-case? robust? domain ns)
        (map (lambda (x) `(check-equal? ,(cons foo x) ',(apply (eval foo ns) x)))
            ((if worst-case? worst-case better-case) (if robust? (robust (BVA domain)) (BVA domain)))))

;
;generateTester turns a specified function into an executable script that tests that function.
;
    (define (generateTester foo worst-case? robust? domain)
        ;Set some local abstractions...
        (let* ( [path (path->string (find-system-path 'run-file))]
            [out (string-append "tests/" (symbol->string foo) (if robust? ".robBVA" ".BVA") (if worst-case? ".wc" ".bc") ".rkt")]
            [ns (module->namespace path)])

            ;If a tester needs to be generated, create it and write its code
            (if (or (not (equal? path (string-replace path "BVA" ""))) (file-exists? out))
                (void)
                (with-output-to-file out
                    (lambda () (begin (printf (string-append "#lang racket\n(require rackunit)\n(require \"../" path "\")\n\n"))
                        (foldl (lambda (x acc) (begin (write x) (printf "\n")))
                            0
                            (generateTestCases foo worst-case? robust? domain ns))))))))
```

Implementation of domain/domain.rkt:

```
#lang racket

(provide (contract-out
        [domain?       (-> any/c boolean?)]
        [BVA           (-> domain? domain?)]
        [robust        (-> domain? domain?)]))

;What qualifies as a domain?
    (define (domain? D) (if (match D [(list (list a b _ ...) ...) #t] [_ #f])
                    (andmap (lambda (d) (andmap integer? d)) D)
                    #f))

;
;Domain transforming functions
;
    ;Turn a list of ranges into a list of boundary value lists
    (define (BVA domain)
        (let ([nom (lambda (x) (round (/ (+ (car x) (last x)) 2)))])
            (map (lambda (arg) (list (car arg) (+ 1 (car arg)) (nom arg) (- (last arg) 1) (last arg))) domain)))

    ;Robustify a domain
    (define (robust domain) (map (lambda (x) (flatten (list (- (car x) 1) x (+ 1 (last x))))) domain))
```

Implementation of permutation/permutation.rkt:

```
#lang racket

(provide (contract-out
        [combination?   (-> any/c boolean?)]
        [cross          (-> any/c list? list?)]
        [crossproduct   (-> list? list? list?)]
        [replace        (-> any/c any/c list? list?)]))

;What qualifies as a combination?
    (define (combination? C) (match C [(list (list _ ...) ...) #t] [_ #f]))

;
;Set combinator functions
;
    ;Calculate a single cross
    (define (cross x ys) (foldl (lambda (y acc) (cons (flatten (list x y)) acc)) '() ys))

    ;Calculate a cross product of two sets
    (define (crossproduct L1 L2) (foldl (lambda (l1 acc) (append (cross l1 L2) acc)) '() L1))

    ;Replace x with y in list L
    (define (replace x y L) (map (lambda (z) (if (equal? x z) y z)) L))
```

Implementation of permutation/domainCases.rkt:

```
#lang racket

(require "permutation.rkt")
(provide (contract-out
	[better-case    (-> combination? combination?)]
	[worst-case     (-> combination? combination?)]))

;
;Auxillary combinator, returns the set of nominals from a domain, not including the nominal of argument x.
;
	(define (otherNoms x xs) (if (pair? xs) (if (equal? x 0)
					;replace x's nominal with 'X, to preserve position
					(cons 'X (otherNoms -1 (cdr xs)))

					;else grab nominal
					(cons (car (drop (car xs) (floor (/ (length (car xs)) 2)))) (otherNoms (- x 1) (cdr xs))))
				null))

;
;Domain combinator functions
;
	;Turn a domain? into its worst-case set of combinations
	(define (worst-case domain)
		;Recursively calculate cross products
		(define (wc d) (if (pair? (cdr d)) (crossproduct (car d) (wc (cdr d))) (car d)))
		(remove-duplicates (wc domain)))

	;Turn a domain? into its single-fault set of combinations
	(define (better-case domain)
		;A partition of the output cases
		(define (portion i x) (cross (otherNoms i domain) x))

		;Remember that parameter position was preserved with 'X marking
		(define (fix-subportions p) (drop-right (replace 'X (last p) p) 1))

		;Create partitions and fix them
		(remove-duplicates (cdr (foldl (lambda (x acc) (cons (+ (car acc) 1) (append (map fix-subportions (portion (car acc) x)) (cdr acc))))
				'(0)
				domain))))
```

Examples

To exemplify use and behavior of the Tennis Racket testing system, this guide includes three reference programs: the Triangle problem, the NextDate problem, and the Commission problem. Each of these three examples will include specification of the problem, a solution expressed in the form of a Racket module, and the associated testers generated by Tennis. In the Triangle problem example, a failed test will be shown for clarification.

The triangle problem:

Given three positive integer inputs a, b, and c, output the type of triangle formed by using a, b, and c as the lengths of its legs. Valid outputs are: “Equilateral”, “Isosceles”, “Scalene”, and “Not a Triangle”. The domain used for this example is: '((5 205) (5 205) (5 205)), and testers will be generated for both the robust worst-case, and the robust better-case.

Triangle problem implementation: ./examples/triangle.rkt

```
#lang racket
(require tennis)
(provide triangle)

(define domain '((5 205) (5 205) (5 205)))

(define (triangle a b c)
      (let ([isATriangle (and (< a (+ b c)) (< b (+ a c)) (< c (+ a b)))])
            (if isATriangle
                  (if (and (= a b) (= b c))
                        "Equilateral"
                        (if (and (not (= a b)) (not (= a c)) (not (= b c)))
                              "Scalene"
                              "Isosceles"))
                  "Not a Triangle")))

(generateTester 'triangle #f #t domain)
(generateTester 'triangle #t #t domain)
```

When executed, this implementation of the Triangle problem will output two tester programs. The first of which is generated as follows at the top of the next page.

Triangle-Tester robust better-case: ./tests/triangle.robBVA.bc.rkt

```
#lang racket
(require rackunit)
(require "../examples/triangle.rkt")

(check-equal? (triangle 105 105 206) (quote "Isosceles"))
(check-equal? (triangle 105 105 205) (quote "Isosceles"))
(check-equal? (triangle 105 105 204) (quote "Isosceles"))
(check-equal? (triangle 105 105 105) (quote "Equilateral"))
(check-equal? (triangle 105 105 6) (quote "Isosceles"))
(check-equal? (triangle 105 105 5) (quote "Isosceles"))
(check-equal? (triangle 105 105 4) (quote "Isosceles"))
(check-equal? (triangle 105 206 105) (quote "Isosceles"))
(check-equal? (triangle 105 205 105) (quote "Isosceles"))
(check-equal? (triangle 105 204 105) (quote "Isosceles"))
(check-equal? (triangle 105 6 105) (quote "Isosceles"))
(check-equal? (triangle 105 5 105) (quote "Isosceles"))
(check-equal? (triangle 105 4 105) (quote "Isosceles"))
(check-equal? (triangle 206 105 105) (quote "Isosceles"))
(check-equal? (triangle 205 105 105) (quote "Isosceles"))
(check-equal? (triangle 204 105 105) (quote "Isosceles"))
(check-equal? (triangle 6 105 105) (quote "Isosceles"))
(check-equal? (triangle 5 105 105) (quote "Isosceles"))
(check-equal? (triangle 4 105 105) (quote "Isosceles"))
```

This triangle tester program can be executed in the same manner as any other Racket program. When it is executed, it will output any failures it encounters. If no errors are found, the program completes with no provided output, as to remain seamless with any project it may be integrated into. The tester program that is originally generated will pass all tests. To see an example of a failure, the above tester program can be modified. When the first line's expected output is changed to “Not a Triangle”, the following output is provided by the tester program:

Triangle Tester failed test example:

```
--------------------
FAILURE
actual:     "Isosceles"
expected:   "Not a Triangle"
name:       check-equal?
location:   (#<path:/home/leftjar/code/tennis/tests/triangle.robBVA.bc.rkt> 5 0 71 54)
expression: (check-equal? (triangle 105 105 206) "Not a Triangle")

Check failure
--------------------
```

Finally, here is the robust worst-case tester program that Tennis generates for the Triangle problem:

Triangle-Tester robust worst-case: /tests/triangle.robBVA.wc.rkt

```
#lang racket
(require rackunit)
(require "../examples/triangle.rkt")

(check-equal? (triangle 206 4 4) (quote "Not a Triangle"))
(check-equal? (triangle 206 4 5) (quote "Not a Triangle"))
(check-equal? (triangle 206 4 6) (quote "Not a Triangle"))
(check-equal? (triangle 206 4 105) (quote "Not a Triangle"))
(check-equal? (triangle 206 4 204) (quote "Scalene"))
(check-equal? (triangle 206 4 205) (quote "Scalene"))
(check-equal? (triangle 206 4 206) (quote "Isosceles"))
(check-equal? (triangle 206 5 4) (quote "Not a Triangle"))
(check-equal? (triangle 206 5 5) (quote "Not a Triangle"))
(check-equal? (triangle 206 5 6) (quote "Not a Triangle"))
(check-equal? (triangle 206 5 105) (quote "Not a Triangle"))
(check-equal? (triangle 206 5 204) (quote "Scalene"))
(check-equal? (triangle 206 5 205) (quote "Scalene"))
(check-equal? (triangle 206 5 206) (quote "Isosceles"))
(check-equal? (triangle 206 6 4) (quote "Not a Triangle"))
(check-equal? (triangle 206 6 5) (quote "Not a Triangle"))
(check-equal? (triangle 206 6 6) (quote "Not a Triangle"))
(check-equal? (triangle 206 6 105) (quote "Not a Triangle"))
(check-equal? (triangle 206 6 204) (quote "Scalene"))
(check-equal? (triangle 206 6 205) (quote "Scalene"))
(check-equal? (triangle 206 6 206) (quote "Isosceles"))
(check-equal? (triangle 206 105 4) (quote "Not a Triangle"))
(check-equal? (triangle 206 105 5) (quote "Not a Triangle"))
(check-equal? (triangle 206 105 6) (quote "Not a Triangle"))
(check-equal? (triangle 206 105 105) (quote "Isosceles"))
(check-equal? (triangle 206 105 204) (quote "Scalene"))
(check-equal? (triangle 206 105 205) (quote "Scalene"))
(check-equal? (triangle 206 105 206) (quote "Isosceles"))
(check-equal? (triangle 206 204 4) (quote "Scalene"))
(check-equal? (triangle 206 204 5) (quote "Scalene"))
(check-equal? (triangle 206 204 6) (quote "Scalene"))
(check-equal? (triangle 206 204 105) (quote "Scalene"))
(check-equal? (triangle 206 204 204) (quote "Isosceles"))
(check-equal? (triangle 206 204 205) (quote "Scalene"))
(check-equal? (triangle 206 204 206) (quote "Isosceles"))
(check-equal? (triangle 206 205 4) (quote "Scalene"))
(check-equal? (triangle 206 205 5) (quote "Scalene"))
(check-equal? (triangle 206 205 6) (quote "Scalene"))
(check-equal? (triangle 206 205 105) (quote "Scalene"))
(check-equal? (triangle 206 205 204) (quote "Scalene"))
(check-equal? (triangle 206 205 205) (quote "Isosceles"))
(check-equal? (triangle 206 205 206) (quote "Isosceles"))
(check-equal? (triangle 206 206 4) (quote "Isosceles"))
(check-equal? (triangle 206 206 5) (quote "Isosceles"))
(check-equal? (triangle 206 206 6) (quote "Isosceles"))
(check-equal? (triangle 206 206 105) (quote "Isosceles"))
(check-equal? (triangle 206 206 204) (quote "Isosceles"))
(check-equal? (triangle 206 206 205) (quote "Isosceles"))
(check-equal? (triangle 206 206 206) (quote "Equilateral"))
(check-equal? (triangle 205 4 4) (quote "Not a Triangle"))
(check-equal? (triangle 205 4 5) (quote "Not a Triangle"))
(check-equal? (triangle 205 4 6) (quote "Not a Triangle"))
(check-equal? (triangle 205 4 105) (quote "Not a Triangle"))
(check-equal? (triangle 205 4 204) (quote "Scalene"))
(check-equal? (triangle 205 4 205) (quote "Isosceles"))
(check-equal? (triangle 205 4 206) (quote "Scalene"))
(check-equal? (triangle 205 5 4) (quote "Not a Triangle"))
(check-equal? (triangle 205 5 5) (quote "Not a Triangle"))
(check-equal? (triangle 205 5 6) (quote "Not a Triangle"))
(check-equal? (triangle 205 5 105) (quote "Not a Triangle"))
(check-equal? (triangle 205 5 204) (quote "Scalene"))
(check-equal? (triangle 205 5 205) (quote "Isosceles"))
(check-equal? (triangle 205 5 206) (quote "Scalene"))
(check-equal? (triangle 205 6 4) (quote "Not a Triangle"))
(check-equal? (triangle 105 105 6) (quote "Isosceles"))
(check-equal? (triangle 105 105 105) (quote "Equilateral"))
(check-equal? (triangle 105 105 204) (quote "Isosceles"))
(check-equal? (triangle 105 105 205) (quote "Isosceles"))
(check-equal? (triangle 105 105 206) (quote "Isosceles"))
(check-equal? (triangle 105 204 4) (quote "Not a Triangle"))
(check-equal? (triangle 105 204 5) (quote "Not a Triangle"))
(check-equal? (triangle 105 204 6) (quote "Not a Triangle"))
(check-equal? (triangle 105 204 105) (quote "Isosceles"))
(check-equal? (triangle 105 204 204) (quote "Isosceles"))
(check-equal? (triangle 105 204 205) (quote "Scalene"))
(check-equal? (triangle 105 204 206) (quote "Scalene"))
(check-equal? (triangle 105 205 4) (quote "Not a Triangle"))
(check-equal? (triangle 105 205 5) (quote "Not a Triangle"))
(check-equal? (triangle 105 205 6) (quote "Not a Triangle"))
(check-equal? (triangle 105 205 105) (quote "Isosceles"))
(check-equal? (triangle 105 205 204) (quote "Scalene"))
(check-equal? (triangle 105 205 205) (quote "Isosceles"))
(check-equal? (triangle 105 205 206) (quote "Scalene"))
(check-equal? (triangle 105 206 4) (quote "Not a Triangle"))
(check-equal? (triangle 105 206 5) (quote "Not a Triangle"))
(check-equal? (triangle 105 206 6) (quote "Not a Triangle"))
(check-equal? (triangle 105 206 105) (quote "Isosceles"))
(check-equal? (triangle 105 206 204) (quote "Scalene"))
(check-equal? (triangle 105 206 205) (quote "Scalene"))
(check-equal? (triangle 105 206 206) (quote "Isosceles"))
(check-equal? (triangle 6 4 4) (quote "Isosceles"))
(check-equal? (triangle 6 4 5) (quote "Scalene"))
(check-equal? (triangle 6 4 6) (quote "Isosceles"))
(check-equal? (triangle 6 4 105) (quote "Not a Triangle"))
(check-equal? (triangle 6 4 204) (quote "Not a Triangle"))
(check-equal? (triangle 6 4 205) (quote "Not a Triangle"))
(check-equal? (triangle 6 4 206) (quote "Not a Triangle"))
(check-equal? (triangle 6 5 4) (quote "Scalene"))
(check-equal? (triangle 6 5 5) (quote "Isosceles"))
(check-equal? (triangle 6 5 6) (quote "Isosceles"))
(check-equal? (triangle 6 5 105) (quote "Not a Triangle"))
(check-equal? (triangle 6 5 204) (quote "Not a Triangle"))
(check-equal? (triangle 6 5 205) (quote "Not a Triangle"))
(check-equal? (triangle 6 5 206) (quote "Not a Triangle"))
(check-equal? (triangle 6 6 4) (quote "Isosceles"))
(check-equal? (triangle 6 6 5) (quote "Isosceles"))
(check-equal? (triangle 6 6 6) (quote "Equilateral"))
(check-equal? (triangle 6 6 105) (quote "Not a Triangle"))
(check-equal? (triangle 6 6 204) (quote "Not a Triangle"))
(check-equal? (triangle 6 6 205) (quote "Not a Triangle"))
(check-equal? (triangle 6 6 206) (quote "Not a Triangle"))
(check-equal? (triangle 6 105 4) (quote "Not a Triangle"))
(check-equal? (triangle 6 105 5) (quote "Not a Triangle"))
(check-equal? (triangle 6 105 6) (quote "Not a Triangle"))
(check-equal? (triangle 6 105 105) (quote "Isosceles"))
(check-equal? (triangle 6 105 204) (quote "Not a Triangle"))
(check-equal? (triangle 6 105 205) (quote "Not a Triangle"))
(check-equal? (triangle 6 105 206) (quote "Not a Triangle"))
(check-equal? (triangle 6 204 4) (quote "Not a Triangle"))
(check-equal? (triangle 6 204 5) (quote "Not a Triangle"))
(check-equal? (triangle 6 204 6) (quote "Not a Triangle"))
(check-equal? (triangle 6 204 105) (quote "Not a Triangle"))
(check-equal? (triangle 6 204 204) (quote "Isosceles"))
(check-equal? (triangle 6 204 205) (quote "Scalene"))
(check-equal? (triangle 6 204 206) (quote "Scalene"))
(check-equal? (triangle 6 205 4) (quote "Not a Triangle"))
(check-equal? (triangle 6 205 5) (quote "Not a Triangle"))
(check-equal? (triangle 6 205 6) (quote "Not a Triangle"))
(check-equal? (triangle 6 205 105) (quote "Not a Triangle"))
(check-equal? (triangle 6 205 204) (quote "Scalene"))
(check-equal? (triangle 6 205 205) (quote "Isosceles"))
(check-equal? (triangle 6 205 206) (quote "Scalene"))
```

```
(check-equal? (triangle 205 6 5) (quote "Not a Triangle"))
(check-equal? (triangle 205 6 6) (quote "Not a Triangle"))
(check-equal? (triangle 205 6 105) (quote "Not a Triangle"))
(check-equal? (triangle 205 6 204) (quote "Scalene"))
(check-equal? (triangle 205 6 205) (quote "Isosceles"))
(check-equal? (triangle 205 6 206) (quote "Scalene"))
(check-equal? (triangle 205 105 4) (quote "Not a Triangle"))
(check-equal? (triangle 205 105 5) (quote "Not a Triangle"))
(check-equal? (triangle 205 105 6) (quote "Not a Triangle"))
(check-equal? (triangle 205 105 105) (quote "Isosceles"))
(check-equal? (triangle 205 105 204) (quote "Scalene"))
(check-equal? (triangle 205 105 205) (quote "Isosceles"))
(check-equal? (triangle 205 105 206) (quote "Scalene"))
(check-equal? (triangle 205 204 4) (quote "Scalene"))
(check-equal? (triangle 205 204 5) (quote "Scalene"))
(check-equal? (triangle 205 204 6) (quote "Scalene"))
(check-equal? (triangle 205 204 105) (quote "Scalene"))
(check-equal? (triangle 205 204 204) (quote "Isosceles"))
(check-equal? (triangle 205 204 205) (quote "Isosceles"))
(check-equal? (triangle 205 204 206) (quote "Scalene"))
(check-equal? (triangle 205 205 4) (quote "Isosceles"))
(check-equal? (triangle 205 205 5) (quote "Isosceles"))
(check-equal? (triangle 205 205 6) (quote "Isosceles"))
(check-equal? (triangle 205 205 105) (quote "Isosceles"))
(check-equal? (triangle 205 205 204) (quote "Isosceles"))
(check-equal? (triangle 205 205 205) (quote "Equilateral"))
(check-equal? (triangle 205 205 206) (quote "Isosceles"))
(check-equal? (triangle 205 206 4) (quote "Scalene"))
(check-equal? (triangle 205 206 5) (quote "Scalene"))
(check-equal? (triangle 205 206 6) (quote "Scalene"))
(check-equal? (triangle 205 206 105) (quote "Scalene"))
(check-equal? (triangle 205 206 204) (quote "Scalene"))
(check-equal? (triangle 205 206 205) (quote "Isosceles"))
(check-equal? (triangle 205 206 206) (quote "Isosceles"))
(check-equal? (triangle 204 4 4) (quote "Not a Triangle"))
(check-equal? (triangle 204 4 5) (quote "Not a Triangle"))
(check-equal? (triangle 204 4 6) (quote "Not a Triangle"))
(check-equal? (triangle 204 4 105) (quote "Not a Triangle"))
(check-equal? (triangle 204 4 204) (quote "Isosceles"))
(check-equal? (triangle 204 4 205) (quote "Scalene"))
(check-equal? (triangle 204 4 206) (quote "Scalene"))
(check-equal? (triangle 204 5 4) (quote "Not a Triangle"))
(check-equal? (triangle 204 5 5) (quote "Not a Triangle"))
(check-equal? (triangle 204 5 6) (quote "Not a Triangle"))
(check-equal? (triangle 204 5 105) (quote "Not a Triangle"))
(check-equal? (triangle 204 5 204) (quote "Isosceles"))
(check-equal? (triangle 204 5 205) (quote "Scalene"))
(check-equal? (triangle 204 5 206) (quote "Scalene"))
(check-equal? (triangle 204 6 4) (quote "Not a Triangle"))
(check-equal? (triangle 204 6 5) (quote "Not a Triangle"))
(check-equal? (triangle 204 6 6) (quote "Not a Triangle"))
(check-equal? (triangle 204 6 105) (quote "Not a Triangle"))
(check-equal? (triangle 204 6 204) (quote "Isosceles"))
(check-equal? (triangle 204 6 205) (quote "Scalene"))
(check-equal? (triangle 204 6 206) (quote "Scalene"))
(check-equal? (triangle 204 105 4) (quote "Not a Triangle"))
(check-equal? (triangle 204 105 5) (quote "Not a Triangle"))
(check-equal? (triangle 204 105 6) (quote "Not a Triangle"))
(check-equal? (triangle 204 105 105) (quote "Isosceles"))
(check-equal? (triangle 204 105 204) (quote "Isosceles"))
(check-equal? (triangle 204 105 205) (quote "Scalene"))
(check-equal? (triangle 204 105 206) (quote "Scalene"))
(check-equal? (triangle 204 204 4) (quote "Isosceles"))
(check-equal? (triangle 204 204 5) (quote "Isosceles"))
(check-equal? (triangle 204 204 6) (quote "Isosceles"))
(check-equal? (triangle 204 204 105) (quote "Isosceles"))
(check-equal? (triangle 204 204 204) (quote "Equilateral"))
(check-equal? (triangle 204 204 205) (quote "Isosceles"))
(check-equal? (triangle 204 204 206) (quote "Isosceles"))
(check-equal? (triangle 204 205 4) (quote "Scalene"))
(check-equal? (triangle 6 206 4) (quote "Not a Triangle"))
(check-equal? (triangle 6 206 5) (quote "Not a Triangle"))
(check-equal? (triangle 6 206 6) (quote "Not a Triangle"))
(check-equal? (triangle 6 206 105) (quote "Not a Triangle"))
(check-equal? (triangle 6 206 204) (quote "Scalene"))
(check-equal? (triangle 6 206 205) (quote "Scalene"))
(check-equal? (triangle 6 206 206) (quote "Isosceles"))
(check-equal? (triangle 5 4 4) (quote "Isosceles"))
(check-equal? (triangle 5 4 5) (quote "Isosceles"))
(check-equal? (triangle 5 4 6) (quote "Scalene"))
(check-equal? (triangle 5 4 105) (quote "Not a Triangle"))
(check-equal? (triangle 5 4 204) (quote "Not a Triangle"))
(check-equal? (triangle 5 4 205) (quote "Not a Triangle"))
(check-equal? (triangle 5 4 206) (quote "Not a Triangle"))
(check-equal? (triangle 5 5 4) (quote "Isosceles"))
(check-equal? (triangle 5 5 5) (quote "Equilateral"))
(check-equal? (triangle 5 5 6) (quote "Isosceles"))
(check-equal? (triangle 5 5 105) (quote "Not a Triangle"))
(check-equal? (triangle 5 5 204) (quote "Not a Triangle"))
(check-equal? (triangle 5 5 205) (quote "Not a Triangle"))
(check-equal? (triangle 5 5 206) (quote "Not a Triangle"))
(check-equal? (triangle 5 6 4) (quote "Scalene"))
(check-equal? (triangle 5 6 5) (quote "Isosceles"))
(check-equal? (triangle 5 6 6) (quote "Isosceles"))
(check-equal? (triangle 5 6 105) (quote "Not a Triangle"))
(check-equal? (triangle 5 6 204) (quote "Not a Triangle"))
(check-equal? (triangle 5 6 205) (quote "Not a Triangle"))
(check-equal? (triangle 5 6 206) (quote "Not a Triangle"))
(check-equal? (triangle 5 105 4) (quote "Not a Triangle"))
(check-equal? (triangle 5 105 5) (quote "Not a Triangle"))
(check-equal? (triangle 5 105 6) (quote "Not a Triangle"))
(check-equal? (triangle 5 105 105) (quote "Isosceles"))
(check-equal? (triangle 5 105 204) (quote "Not a Triangle"))
(check-equal? (triangle 5 105 205) (quote "Not a Triangle"))
(check-equal? (triangle 5 105 206) (quote "Not a Triangle"))
(check-equal? (triangle 5 204 4) (quote "Not a Triangle"))
(check-equal? (triangle 5 204 5) (quote "Not a Triangle"))
(check-equal? (triangle 5 204 6) (quote "Not a Triangle"))
(check-equal? (triangle 5 204 105) (quote "Not a Triangle"))
(check-equal? (triangle 5 204 204) (quote "Isosceles"))
(check-equal? (triangle 5 204 205) (quote "Scalene"))
(check-equal? (triangle 5 204 206) (quote "Scalene"))
(check-equal? (triangle 5 205 4) (quote "Not a Triangle"))
(check-equal? (triangle 5 205 5) (quote "Not a Triangle"))
(check-equal? (triangle 5 205 6) (quote "Not a Triangle"))
(check-equal? (triangle 5 205 105) (quote "Not a Triangle"))
(check-equal? (triangle 5 205 204) (quote "Scalene"))
(check-equal? (triangle 5 205 205) (quote "Isosceles"))
(check-equal? (triangle 5 205 206) (quote "Scalene"))
(check-equal? (triangle 5 206 4) (quote "Not a Triangle"))
(check-equal? (triangle 5 206 5) (quote "Not a Triangle"))
(check-equal? (triangle 5 206 6) (quote "Not a Triangle"))
(check-equal? (triangle 5 206 105) (quote "Not a Triangle"))
(check-equal? (triangle 5 206 204) (quote "Scalene"))
(check-equal? (triangle 5 206 205) (quote "Scalene"))
(check-equal? (triangle 5 206 206) (quote "Isosceles"))
(check-equal? (triangle 4 4 4) (quote "Equilateral"))
(check-equal? (triangle 4 4 5) (quote "Isosceles"))
(check-equal? (triangle 4 4 6) (quote "Isosceles"))
(check-equal? (triangle 4 4 105) (quote "Not a Triangle"))
(check-equal? (triangle 4 4 204) (quote "Not a Triangle"))
(check-equal? (triangle 4 4 205) (quote "Not a Triangle"))
(check-equal? (triangle 4 4 206) (quote "Not a Triangle"))
(check-equal? (triangle 4 5 4) (quote "Isosceles"))
(check-equal? (triangle 4 5 5) (quote "Isosceles"))
(check-equal? (triangle 4 5 6) (quote "Scalene"))
(check-equal? (triangle 4 5 105) (quote "Not a Triangle"))
(check-equal? (triangle 4 5 204) (quote "Not a Triangle"))
(check-equal? (triangle 4 5 205) (quote "Not a Triangle"))
(check-equal? (triangle 4 5 206) (quote "Not a Triangle"))
```

```
(check-equal? (triangle 204 205 5) (quote "Scalene"))
(check-equal? (triangle 204 205 6) (quote "Scalene"))
(check-equal? (triangle 204 205 105) (quote "Scalene"))
(check-equal? (triangle 204 205 204) (quote "Isosceles"))
(check-equal? (triangle 204 205 205) (quote "Isosceles"))
(check-equal? (triangle 204 205 206) (quote "Scalene"))
(check-equal? (triangle 204 206 4) (quote "Scalene"))
(check-equal? (triangle 204 206 5) (quote "Scalene"))
(check-equal? (triangle 204 206 6) (quote "Scalene"))
(check-equal? (triangle 204 206 105) (quote "Scalene"))
(check-equal? (triangle 204 206 204) (quote "Isosceles"))
(check-equal? (triangle 204 206 205) (quote "Scalene"))
(check-equal? (triangle 204 206 206) (quote "Isosceles"))
(check-equal? (triangle 105 4 4) (quote "Not a Triangle"))
(check-equal? (triangle 105 4 5) (quote "Not a Triangle"))
(check-equal? (triangle 105 4 6) (quote "Not a Triangle"))
(check-equal? (triangle 105 4 105) (quote "Isosceles"))
(check-equal? (triangle 105 4 204) (quote "Not a Triangle"))
(check-equal? (triangle 105 4 205) (quote "Not a Triangle"))
(check-equal? (triangle 105 4 206) (quote "Not a Triangle"))
(check-equal? (triangle 105 5 4) (quote "Not a Triangle"))
(check-equal? (triangle 105 5 5) (quote "Not a Triangle"))
(check-equal? (triangle 105 5 6) (quote "Not a Triangle"))
(check-equal? (triangle 105 5 105) (quote "Isosceles"))
(check-equal? (triangle 105 5 204) (quote "Not a Triangle"))
(check-equal? (triangle 105 5 205) (quote "Not a Triangle"))
(check-equal? (triangle 105 5 206) (quote "Not a Triangle"))
(check-equal? (triangle 105 6 4) (quote "Not a Triangle"))
(check-equal? (triangle 105 6 5) (quote "Not a Triangle"))
(check-equal? (triangle 105 6 6) (quote "Not a Triangle"))
(check-equal? (triangle 105 6 105) (quote "Isosceles"))
(check-equal? (triangle 105 6 204) (quote "Not a Triangle"))
(check-equal? (triangle 105 6 205) (quote "Not a Triangle"))
(check-equal? (triangle 105 6 206) (quote "Not a Triangle"))
(check-equal? (triangle 105 105 4) (quote "Isosceles"))
(check-equal? (triangle 105 105 5) (quote "Isosceles"))
(check-equal? (triangle 4 6 4) (quote "Isosceles"))
(check-equal? (triangle 4 6 5) (quote "Scalene"))
(check-equal? (triangle 4 6 6) (quote "Isosceles"))
(check-equal? (triangle 4 6 105) (quote "Not a Triangle"))
(check-equal? (triangle 4 6 204) (quote "Not a Triangle"))
(check-equal? (triangle 4 6 205) (quote "Not a Triangle"))
(check-equal? (triangle 4 6 206) (quote "Not a Triangle"))
(check-equal? (triangle 4 105 4) (quote "Not a Triangle"))
(check-equal? (triangle 4 105 5) (quote "Not a Triangle"))
(check-equal? (triangle 4 105 6) (quote "Not a Triangle"))
(check-equal? (triangle 4 105 105) (quote "Isosceles"))
(check-equal? (triangle 4 105 204) (quote "Not a Triangle"))
(check-equal? (triangle 4 105 205) (quote "Not a Triangle"))
(check-equal? (triangle 4 105 206) (quote "Not a Triangle"))
(check-equal? (triangle 4 204 4) (quote "Not a Triangle"))
(check-equal? (triangle 4 204 5) (quote "Not a Triangle"))
(check-equal? (triangle 4 204 6) (quote "Not a Triangle"))
(check-equal? (triangle 4 204 105) (quote "Not a Triangle"))
(check-equal? (triangle 4 204 204) (quote "Isosceles"))
(check-equal? (triangle 4 204 205) (quote "Scalene"))
(check-equal? (triangle 4 204 206) (quote "Scalene"))
(check-equal? (triangle 4 205 4) (quote "Not a Triangle"))
(check-equal? (triangle 4 205 5) (quote "Not a Triangle"))
(check-equal? (triangle 4 205 6) (quote "Not a Triangle"))
(check-equal? (triangle 4 205 105) (quote "Not a Triangle"))
(check-equal? (triangle 4 205 204) (quote "Scalene"))
(check-equal? (triangle 4 205 205) (quote "Isosceles"))
(check-equal? (triangle 4 205 206) (quote "Scalene"))
(check-equal? (triangle 4 206 4) (quote "Not a Triangle"))
(check-equal? (triangle 4 206 5) (quote "Not a Triangle"))
(check-equal? (triangle 4 206 6) (quote "Not a Triangle"))
(check-equal? (triangle 4 206 105) (quote "Not a Triangle"))
(check-equal? (triangle 4 206 204) (quote "Scalene"))
(check-equal? (triangle 4 206 205) (quote "Scalene"))
(check-equal? (triangle 4 206 206) (quote "Isosceles"))
```

The NextDate problem:

Given a month M, a day D, and a year Y, output the month, day, and year that immediately follows M/D/Y on a calendar. If no such date exists, output “Invalid Input”. The domain fitted to this function is '((1 12) (1 31) (1801 2014)). Tester programs will be generated over this problem for the robust worst-case, and the robust better-case.

NextDate problem implementation: ./examples/nextdate.rkt

```
#lang racket
(provide nextdate)
(require tennis)

(define domain '((1 12) (1 31) (1801 2014)))
(define (leapyear year) (and (= 0 (modulo year 4)) (not (= 0 (modulo year 400)))))

(define (nextdate month day year)
      (let ([c1 (and (<= 1 day) (<= day 31))]
            [c2 (and (<= 1 month) (<= month 12))]
            [c3 (and (<= 1801 year) (<= year 2014))]
            [tomorrowDay 1] [tomorrowMonth month] [tomorrowYear year])
                  (if (not c1) "Value of day not in range 1..31"
                  (if (not c2) "Value of month not in range 1..12"
                  (if (not c3) "Value of year not in range 1801..2014"
                  (car (filter (lambda (x) (not (void? x))) (list
                        (cond
                              [(member month '(1 3 5 7 8 10)) (if (< day 31) (set! tomorrowDay (+ day 1)) (set! tomorrowMonth (+ month 1)))]
                              [(member month '(4 6 9 11)) (if (< day 30)
                                                (set! tomorrowDay (+ day 1))
                                                (if (= day 30) (set! tomorrowMonth (+ month 1)) "Invalid Input"))]
                              [(= month 12) (if (< day 31) (set! tomorrowDay (+ day 1)) (if (= year 2014)
                                                                  "Invalid Input"
                                                                  (set!-values (tomorrowYear tomorrowMonth) (values (+ year 1) 1))))]
                              [(= month 2) (if (< day 28)
                                          (set! tomorrowDay (+ day 1))
                                          (if (= day 28) (if (leapyear year)
                                                      (set! tomorrowDay 29)
                                                      (set! tomorrowMonth 3))
                                                (if (and (leapyear year) (= day 29))
                                                      (set! tomorrowMonth 3)
                                                      "Invalid Input"))))])
                        (list tomorrowMonth tomorrowDay tomorrowYear)))))))))


(generateTester 'nextdate #f #t domain)
(generateTester 'nextdate #t #t domain)
```

The generateTester calls found in the implementation of the NextDate function generate tester programs as follows:

NextDate-Tester robust better-case: /tests/nextdate.robBVA.bc.rkt

```
#lang racket
(require rackunit)
(require "../examples/nextdate.rkt")

(check-equal? (nextdate 6 16 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 16 2014) (quote (6 17 2014)))
(check-equal? (nextdate 6 16 2013) (quote (6 17 2013)))
(check-equal? (nextdate 6 16 1908) (quote (6 17 1908)))
(check-equal? (nextdate 6 16 1802) (quote (6 17 1802)))
(check-equal? (nextdate 6 16 1801) (quote (6 17 1801)))
(check-equal? (nextdate 6 16 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 31 1908) (quote "Invalid Input"))
(check-equal? (nextdate 6 30 1908) (quote (7 1 1908)))
(check-equal? (nextdate 6 2 1908) (quote (6 3 1908)))
(check-equal? (nextdate 6 1 1908) (quote (6 2 1908)))
(check-equal? (nextdate 6 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 16 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 12 16 1908) (quote (12 17 1908)))
(check-equal? (nextdate 11 16 1908) (quote (11 17 1908)))
(check-equal? (nextdate 2 16 1908) (quote (2 17 1908)))
(check-equal? (nextdate 1 16 1908) (quote (1 17 1908)))
(check-equal? (nextdate 0 16 1908) (quote "Value of month not in range 1..12"))
```

NextDate-Tester robust worst-case: /tests/nextdate.robBVA.wc.rkt

```
#lang racket
(require rackunit)
(require "../examples/nextdate.rkt")

(check-equal? (nextdate 13 0 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 0 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 0 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 0 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 0 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 0 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 1 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 1 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 1 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 1 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 1 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 1 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 1 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 2 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 2 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 2 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 2 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 2 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 2 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 2 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 16 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 16 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 16 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 16 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 16 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 16 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 16 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 30 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 30 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 6 16 1802) (quote (6 17 1802)))
(check-equal? (nextdate 6 16 1908) (quote (6 17 1908)))
(check-equal? (nextdate 6 16 2013) (quote (6 17 2013)))
(check-equal? (nextdate 6 16 2014) (quote (6 17 2014)))
(check-equal? (nextdate 6 16 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 30 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 30 1801) (quote (7 1 1801)))
(check-equal? (nextdate 6 30 1802) (quote (7 1 1802)))
(check-equal? (nextdate 6 30 1908) (quote (7 1 1908)))
(check-equal? (nextdate 6 30 2013) (quote (7 1 2013)))
(check-equal? (nextdate 6 30 2014) (quote (7 1 2014)))
(check-equal? (nextdate 6 30 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 31 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 31 1801) (quote "Invalid Input"))
(check-equal? (nextdate 6 31 1802) (quote "Invalid Input"))
(check-equal? (nextdate 6 31 1908) (quote "Invalid Input"))
(check-equal? (nextdate 6 31 2013) (quote "Invalid Input"))
(check-equal? (nextdate 6 31 2014) (quote "Invalid Input"))
(check-equal? (nextdate 6 31 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 32 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 32 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 32 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 32 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 32 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 32 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 0 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 0 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 0 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 0 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 0 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 0 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 1 1800) (quote "Value of year not in range 1801..2014"))
```

```
(check-equal? (nextdate 13 30 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 30 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 30 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 30 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 30 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 31 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 31 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 31 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 31 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 31 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 31 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 31 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 13 32 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 32 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 32 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 32 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 32 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 13 32 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 0 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 0 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 0 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 0 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 0 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 0 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 1 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 1 1801) (quote (12 2 1801)))
(check-equal? (nextdate 12 1 1802) (quote (12 2 1802)))
(check-equal? (nextdate 12 1 1908) (quote (12 2 1908)))
(check-equal? (nextdate 12 1 2013) (quote (12 2 2013)))
(check-equal? (nextdate 12 1 2014) (quote (12 2 2014)))
(check-equal? (nextdate 12 1 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 2 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 2 1801) (quote (12 3 1801)))
(check-equal? (nextdate 12 2 1802) (quote (12 3 1802)))
(check-equal? (nextdate 12 2 1908) (quote (12 3 1908)))
(check-equal? (nextdate 12 2 2013) (quote (12 3 2013)))
(check-equal? (nextdate 12 2 2014) (quote (12 3 2014)))
(check-equal? (nextdate 12 2 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 16 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 16 1801) (quote (12 17 1801)))
(check-equal? (nextdate 12 16 1802) (quote (12 17 1802)))
(check-equal? (nextdate 12 16 1908) (quote (12 17 1908)))
(check-equal? (nextdate 12 16 2013) (quote (12 17 2013)))
(check-equal? (nextdate 12 16 2014) (quote (12 17 2014)))
(check-equal? (nextdate 12 16 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 30 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 30 1801) (quote (12 31 1801)))
(check-equal? (nextdate 12 30 1802) (quote (12 31 1802)))
(check-equal? (nextdate 12 30 1908) (quote (12 31 1908)))
(check-equal? (nextdate 12 30 2013) (quote (12 31 2013)))
(check-equal? (nextdate 12 30 2014) (quote (12 31 2014)))
(check-equal? (nextdate 12 30 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 31 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 31 1801) (quote (1 1 1802)))
(check-equal? (nextdate 12 31 1802) (quote (1 1 1803)))
(check-equal? (nextdate 12 31 1908) (quote (1 1 1909)))
(check-equal? (nextdate 12 31 2013) (quote (1 1 2014)))
(check-equal? (nextdate 12 31 2014) (quote "Invalid Input"))
(check-equal? (nextdate 12 31 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 12 32 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 32 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 32 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 32 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 32 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 12 32 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 0 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 0 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 1 1801) (quote (2 2 1801)))
(check-equal? (nextdate 2 1 1802) (quote (2 2 1802)))
(check-equal? (nextdate 2 1 1908) (quote (2 2 1908)))
(check-equal? (nextdate 2 1 2013) (quote (2 2 2013)))
(check-equal? (nextdate 2 1 2014) (quote (2 2 2014)))
(check-equal? (nextdate 2 1 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 2 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 2 1801) (quote (2 3 1801)))
(check-equal? (nextdate 2 2 1802) (quote (2 3 1802)))
(check-equal? (nextdate 2 2 1908) (quote (2 3 1908)))
(check-equal? (nextdate 2 2 2013) (quote (2 3 2013)))
(check-equal? (nextdate 2 2 2014) (quote (2 3 2014)))
(check-equal? (nextdate 2 2 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 16 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 16 1801) (quote (2 17 1801)))
(check-equal? (nextdate 2 16 1802) (quote (2 17 1802)))
(check-equal? (nextdate 2 16 1908) (quote (2 17 1908)))
(check-equal? (nextdate 2 16 2013) (quote (2 17 2013)))
(check-equal? (nextdate 2 16 2014) (quote (2 17 2014)))
(check-equal? (nextdate 2 16 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 30 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 30 1801) (quote "Invalid Input"))
(check-equal? (nextdate 2 30 1802) (quote "Invalid Input"))
(check-equal? (nextdate 2 30 1908) (quote "Invalid Input"))
(check-equal? (nextdate 2 30 2013) (quote "Invalid Input"))
(check-equal? (nextdate 2 30 2014) (quote "Invalid Input"))
(check-equal? (nextdate 2 30 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 31 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 31 1801) (quote "Invalid Input"))
(check-equal? (nextdate 2 31 1802) (quote "Invalid Input"))
(check-equal? (nextdate 2 31 1908) (quote "Invalid Input"))
(check-equal? (nextdate 2 31 2013) (quote "Invalid Input"))
(check-equal? (nextdate 2 31 2014) (quote "Invalid Input"))
(check-equal? (nextdate 2 31 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 2 32 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 32 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 32 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 32 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 32 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 2 32 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 0 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 0 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 0 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 0 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 0 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 0 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 1 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 1 1801) (quote (1 2 1801)))
(check-equal? (nextdate 1 1 1802) (quote (1 2 1802)))
(check-equal? (nextdate 1 1 1908) (quote (1 2 1908)))
(check-equal? (nextdate 1 1 2013) (quote (1 2 2013)))
(check-equal? (nextdate 1 1 2014) (quote (1 2 2014)))
(check-equal? (nextdate 1 1 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 2 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 2 1801) (quote (1 3 1801)))
(check-equal? (nextdate 1 2 1802) (quote (1 3 1802)))
(check-equal? (nextdate 1 2 1908) (quote (1 3 1908)))
(check-equal? (nextdate 1 2 2013) (quote (1 3 2013)))
(check-equal? (nextdate 1 2 2014) (quote (1 3 2014)))
(check-equal? (nextdate 1 2 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 16 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 16 1801) (quote (1 17 1801)))
(check-equal? (nextdate 1 16 1802) (quote (1 17 1802)))
(check-equal? (nextdate 1 16 1908) (quote (1 17 1908)))
(check-equal? (nextdate 1 16 2013) (quote (1 17 2013)))
(check-equal? (nextdate 1 16 2014) (quote (1 17 2014)))
(check-equal? (nextdate 1 16 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 30 1800) (quote "Value of year not in range 1801..2014"))
```

```
(check-equal? (nextdate 11 0 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 0 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 0 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 0 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 1 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 1 1801) (quote (11 2 1801)))
(check-equal? (nextdate 11 1 1802) (quote (11 2 1802)))
(check-equal? (nextdate 11 1 1908) (quote (11 2 1908)))
(check-equal? (nextdate 11 1 2013) (quote (11 2 2013)))
(check-equal? (nextdate 11 1 2014) (quote (11 2 2014)))
(check-equal? (nextdate 11 1 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 2 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 2 1801) (quote (11 3 1801)))
(check-equal? (nextdate 11 2 1802) (quote (11 3 1802)))
(check-equal? (nextdate 11 2 1908) (quote (11 3 1908)))
(check-equal? (nextdate 11 2 2013) (quote (11 3 2013)))
(check-equal? (nextdate 11 2 2014) (quote (11 3 2014)))
(check-equal? (nextdate 11 2 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 16 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 16 1801) (quote (11 17 1801)))
(check-equal? (nextdate 11 16 1802) (quote (11 17 1802)))
(check-equal? (nextdate 11 16 1908) (quote (11 17 1908)))
(check-equal? (nextdate 11 16 2013) (quote (11 17 2013)))
(check-equal? (nextdate 11 16 2014) (quote (11 17 2014)))
(check-equal? (nextdate 11 16 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 30 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 30 1801) (quote (12 1 1801)))
(check-equal? (nextdate 11 30 1802) (quote (12 1 1802)))
(check-equal? (nextdate 11 30 1908) (quote (12 1 1908)))
(check-equal? (nextdate 11 30 2013) (quote (12 1 2013)))
(check-equal? (nextdate 11 30 2014) (quote (12 1 2014)))
(check-equal? (nextdate 11 30 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 31 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 31 1801) (quote "Invalid Input"))
(check-equal? (nextdate 11 31 1802) (quote "Invalid Input"))
(check-equal? (nextdate 11 31 1908) (quote "Invalid Input"))
(check-equal? (nextdate 11 31 2013) (quote "Invalid Input"))
(check-equal? (nextdate 11 31 2014) (quote "Invalid Input"))
(check-equal? (nextdate 11 31 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 11 32 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 32 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 32 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 32 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 32 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 11 32 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 0 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 0 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 0 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 0 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 0 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 0 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 6 1 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 1 1801) (quote (6 2 1801)))
(check-equal? (nextdate 6 1 1802) (quote (6 2 1802)))
(check-equal? (nextdate 6 1 1908) (quote (6 2 1908)))
(check-equal? (nextdate 6 1 2013) (quote (6 2 2013)))
(check-equal? (nextdate 6 1 2014) (quote (6 2 2014)))
(check-equal? (nextdate 6 1 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 2 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 2 1801) (quote (6 3 1801)))
(check-equal? (nextdate 6 2 1802) (quote (6 3 1802)))
(check-equal? (nextdate 6 2 1908) (quote (6 3 1908)))
(check-equal? (nextdate 6 2 2013) (quote (6 3 2013)))
(check-equal? (nextdate 6 2 2014) (quote (6 3 2014)))
(check-equal? (nextdate 6 2 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 16 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 6 16 1801) (quote (6 17 1801)))
(check-equal? (nextdate 1 30 1801) (quote (1 31 1801)))
(check-equal? (nextdate 1 30 1802) (quote (1 31 1802)))
(check-equal? (nextdate 1 30 1908) (quote (1 31 1908)))
(check-equal? (nextdate 1 30 2013) (quote (1 31 2013)))
(check-equal? (nextdate 1 30 2014) (quote (1 31 2014)))
(check-equal? (nextdate 1 30 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 31 1800) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 31 1801) (quote (2 1 1801)))
(check-equal? (nextdate 1 31 1802) (quote (2 1 1802)))
(check-equal? (nextdate 1 31 1908) (quote (2 1 1908)))
(check-equal? (nextdate 1 31 2013) (quote (2 1 2013)))
(check-equal? (nextdate 1 31 2014) (quote (2 1 2014)))
(check-equal? (nextdate 1 31 2015) (quote "Value of year not in range 1801..2014"))
(check-equal? (nextdate 1 32 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 32 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 32 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 32 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 32 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 1 32 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 0 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 0 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 0 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 0 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 0 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 0 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 0 2015) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 1 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 1 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 1 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 1 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 1 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 1 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 1 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 2 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 2 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 2 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 2 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 2 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 2 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 2 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 16 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 16 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 16 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 16 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 16 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 16 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 16 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 30 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 30 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 30 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 30 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 30 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 30 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 30 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 31 1800) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 31 1801) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 31 1802) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 31 1908) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 31 2013) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 31 2014) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 31 2015) (quote "Value of month not in range 1..12"))
(check-equal? (nextdate 0 32 1800) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 32 1801) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 32 1802) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 32 1908) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 32 2013) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 32 2014) (quote "Value of day not in range 1..31"))
(check-equal? (nextdate 0 32 2015) (quote "Value of day not in range 1..31"))
```

The Commission problem:

Given a number of locks L, stocks S, and barrels B sold, output the commission owed to the salesman. Each lock costs $35, each stock $25, and each barrel $20. For the first $1000 sold, a commission of 10% is owed. For the next $800, 15% commission is owed. A commission of 20% is owed for all profit over $1800. The domain representing valid ranges for L, S, and B is '((0 77) (0 88) (0 99)). Tester programs will be generated over this problem for the non-robust better-case, robust better-case, and the non-robust worst-case.

Commission problem implementation: ./examples/commission.rkt

```
#lang racket
(provide commission)
(require tennis)

(define domain '((0 77) (0 88) (0 99)))

(define (between x X) (and (<= (car X) x) (>= (cadr X) x)))

(define lockPrice 35)
(define stockPrice 25)
(define barrelPrice 20)

(define (commission locks stocks barrels)
      (let* ( [lockSales (* locks lockPrice)]
            [stockSales (* stocks stockPrice)]
            [barrelSales (* barrels barrelPrice)]
            [sales (+ lockSales stockSales barrelSales)])

            (if (and (between locks (car domain)) (between stocks (cadr domain)) (between barrels (caddr domain)))
                  (if (> sales 1800) (+ 100 120 (* .2 (- sales 1800)))
                        (if (> sales 1000) (+ 100 (* .15 (- sales 1000)))
                        (* .1 sales)))
                  "Invalid input.")))


(generateTester 'commission #f #f domain)
(generateTester 'commission #f #t domain)
(generateTester 'commission #t #f domain)
```

The three generateTester calls placed in the Commission problem's module generate the following tester programs:

Commission-Tester better-case BVA: /tests/commission.BVA.bc.rkt

```
#lang racket
(require rackunit)
(require "../examples/commission.rkt")

(check-equal? (commission 38 44 99) (quote 742.0))
(check-equal? (commission 38 44 98) (quote 738.0))
(check-equal? (commission 38 44 50) (quote 546.0))
(check-equal? (commission 38 44 1) (quote 350.0))
(check-equal? (commission 38 44 0) (quote 346.0))
(check-equal? (commission 38 88 50) (quote 766.0))
(check-equal? (commission 38 87 50) (quote 761.0))
(check-equal? (commission 38 1 50) (quote 331.0))
(check-equal? (commission 38 0 50) (quote 326.0))
(check-equal? (commission 77 44 50) (quote 819.0))
(check-equal? (commission 76 44 50) (quote 812.0))
(check-equal? (commission 1 44 50) (quote 287.0))
(check-equal? (commission 0 44 50) (quote 280.0))
```

Commission-Tester robust better-case: /tests/commission.robBVA.bc.rkt

```
#lang racket
(require rackunit)
(require "../examples/commission.rkt")

(check-equal? (commission 38 44 100) (quote "Invalid input."))
(check-equal? (commission 38 44 99) (quote 742.0))
(check-equal? (commission 38 44 98) (quote 738.0))
(check-equal? (commission 38 44 50) (quote 546.0))
(check-equal? (commission 38 44 1) (quote 350.0))
(check-equal? (commission 38 44 0) (quote 346.0))
(check-equal? (commission 38 44 -1) (quote "Invalid input."))
(check-equal? (commission 38 89 50) (quote "Invalid input."))
(check-equal? (commission 38 88 50) (quote 766.0))
(check-equal? (commission 38 87 50) (quote 761.0))
(check-equal? (commission 38 1 50) (quote 331.0))
(check-equal? (commission 38 0 50) (quote 326.0))
(check-equal? (commission 38 -1 50) (quote "Invalid input."))
(check-equal? (commission 78 44 50) (quote "Invalid input."))
(check-equal? (commission 77 44 50) (quote 819.0))
(check-equal? (commission 76 44 50) (quote 812.0))
(check-equal? (commission 1 44 50) (quote 287.0))
(check-equal? (commission 0 44 50) (quote 280.0))
(check-equal? (commission -1 44 50) (quote "Invalid input."))
```

Commission-Tester worst-case BVA: /tests/commission.BVA.wc.rkt

```
#lang racket
(require rackunit)
(require "../examples/commission.rkt")

(check-equal? (commission 77 0 0) (quote 399.0))
(check-equal? (commission 77 0 1) (quote 403.0))
(check-equal? (commission 77 0 50) (quote 599.0))
(check-equal? (commission 77 0 98) (quote 791.0))
(check-equal? (commission 77 0 99) (quote 795.0))
(check-equal? (commission 77 1 0) (quote 404.0))
(check-equal? (commission 77 1 1) (quote 408.0))
(check-equal? (commission 77 1 50) (quote 604.0))
(check-equal? (commission 77 1 98) (quote 796.0))
(check-equal? (commission 77 1 99) (quote 800.0))
(check-equal? (commission 77 44 0) (quote 619.0))
(check-equal? (commission 77 44 1) (quote 623.0))
(check-equal? (commission 77 44 50) (quote 819.0))
(check-equal? (commission 77 44 98) (quote 1011.0))
(check-equal? (commission 77 44 99) (quote 1015.0))
(check-equal? (commission 77 87 0) (quote 834.0))
(check-equal? (commission 77 87 1) (quote 838.0))
(check-equal? (commission 77 87 50) (quote 1034.0))
(check-equal? (commission 77 87 98) (quote 1226.0))
(check-equal? (commission 77 87 99) (quote 1230.0))
(check-equal? (commission 77 88 0) (quote 839.0))
(check-equal? (commission 77 88 1) (quote 843.0))
(check-equal? (commission 77 88 50) (quote 1039.0))
(check-equal? (commission 77 88 98) (quote 1231.0))
(check-equal? (commission 77 88 99) (quote 1235.0))
(check-equal? (commission 76 0 0) (quote 392.0))
(check-equal? (commission 76 0 1) (quote 396.0))
(check-equal? (commission 76 0 50) (quote 592.0))
(check-equal? (commission 76 0 98) (quote 784.0))
(check-equal? (commission 76 0 99) (quote 788.0))
(check-equal? (commission 76 1 0) (quote 397.0))
(check-equal? (commission 76 1 1) (quote 401.0))
(check-equal? (commission 76 1 50) (quote 597.0))
(check-equal? (commission 76 1 98) (quote 789.0))
(check-equal? (commission 76 1 99) (quote 793.0))
(check-equal? (commission 76 44 0) (quote 612.0))
(check-equal? (commission 76 44 1) (quote 616.0))
(check-equal? (commission 76 44 50) (quote 812.0))
(check-equal? (commission 76 44 98) (quote 1004.0))
(check-equal? (commission 76 44 99) (quote 1008.0))
(check-equal? (commission 76 87 0) (quote 827.0))
(check-equal? (commission 76 87 1) (quote 831.0))
(check-equal? (commission 76 87 50) (quote 1027.0))
(check-equal? (commission 76 87 98) (quote 1219.0))
(check-equal? (commission 76 87 99) (quote 1223.0))
(check-equal? (commission 76 88 0) (quote 832.0))
(check-equal? (commission 76 88 1) (quote 836.0))
(check-equal? (commission 76 88 50) (quote 1032.0))
(check-equal? (commission 76 88 98) (quote 1224.0))
(check-equal? (commission 76 88 99) (quote 1228.0))
(check-equal? (commission 38 0 0) (quote 149.5))
(check-equal? (commission 38 0 1) (quote 152.5))
(check-equal? (commission 38 0 50) (quote 326.0))
(check-equal? (commission 38 0 98) (quote 518.0))
(check-equal? (commission 38 0 99) (quote 522.0))
(check-equal? (commission 38 1 0) (quote 153.25))
(check-equal? (commission 38 1 1) (quote 156.25))
(check-equal? (commission 38 1 50) (quote 331.0))
(check-equal? (commission 38 1 98) (quote 523.0))
(check-equal? (commission 38 1 99) (quote 527.0))
(check-equal? (commission 38 44 0) (quote 346.0))
(check-equal? (commission 38 44 1) (quote 350.0))
(check-equal? (commission 38 44 50) (quote 546.0))
(check-equal? (commission 38 44 98) (quote 738.0))
(check-equal? (commission 38 44 99) (quote 742.0))
(check-equal? (commission 38 87 0) (quote 561.0))
(check-equal? (commission 38 87 1) (quote 565.0))
(check-equal? (commission 38 87 50) (quote 761.0))
(check-equal? (commission 38 87 98) (quote 953.0))
(check-equal? (commission 38 87 99) (quote 957.0))
(check-equal? (commission 38 88 0) (quote 566.0))
(check-equal? (commission 38 88 1) (quote 570.0))
(check-equal? (commission 38 88 50) (quote 766.0))
(check-equal? (commission 38 88 98) (quote 958.0))
(check-equal? (commission 38 88 99) (quote 962.0))
(check-equal? (commission 1 0 0) (quote 3.5))
(check-equal? (commission 1 0 1) (quote 5.5))
(check-equal? (commission 1 0 50) (quote 105.25))
(check-equal? (commission 1 0 98) (quote 259.0))
(check-equal? (commission 1 0 99) (quote 263.0))
(check-equal? (commission 1 1 0) (quote 6.0))
(check-equal? (commission 1 1 1) (quote 8.0))
(check-equal? (commission 1 1 50) (quote 109.0))
(check-equal? (commission 1 1 98) (quote 264.0))
(check-equal? (commission 1 1 99) (quote 268.0))
(check-equal? (commission 1 44 0) (quote 120.25))
(check-equal? (commission 1 44 1) (quote 123.25))
(check-equal? (commission 1 44 50) (quote 287.0))
(check-equal? (commission 1 44 98) (quote 479.0))
(check-equal? (commission 1 44 99) (quote 483.0))
(check-equal? (commission 1 87 0) (quote 302.0))
(check-equal? (commission 1 87 1) (quote 306.0))
(check-equal? (commission 1 87 50) (quote 502.0))
(check-equal? (commission 1 87 98) (quote 694.0))
(check-equal? (commission 1 87 99) (quote 698.0))
(check-equal? (commission 1 88 0) (quote 307.0))
(check-equal? (commission 1 88 1) (quote 311.0))
(check-equal? (commission 1 88 50) (quote 507.0))
(check-equal? (commission 1 88 98) (quote 699.0))
(check-equal? (commission 1 88 99) (quote 703.0))
(check-equal? (commission 0 0 0) (quote 0))
(check-equal? (commission 0 0 1) (quote 2.0))
(check-equal? (commission 0 0 50) (quote 100.0))
(check-equal? (commission 0 0 98) (quote 252.0))
(check-equal? (commission 0 0 99) (quote 256.0))
(check-equal? (commission 0 1 0) (quote 2.5))
(check-equal? (commission 0 1 1) (quote 4.5))
(check-equal? (commission 0 1 50) (quote 103.75))
(check-equal? (commission 0 1 98) (quote 257.0))
(check-equal? (commission 0 1 99) (quote 261.0))
(check-equal? (commission 0 44 0) (quote 115.0))
(check-equal? (commission 0 44 1) (quote 118.0))
(check-equal? (commission 0 44 50) (quote 280.0))
(check-equal? (commission 0 44 98) (quote 472.0))
(check-equal? (commission 0 44 99) (quote 476.0))
(check-equal? (commission 0 87 0) (quote 295.0))
(check-equal? (commission 0 87 1) (quote 299.0))
(check-equal? (commission 0 87 50) (quote 495.0))
(check-equal? (commission 0 87 98) (quote 687.0))
(check-equal? (commission 0 87 99) (quote 691.0))
(check-equal? (commission 0 88 0) (quote 300.0))
(check-equal? (commission 0 88 1) (quote 304.0))
(check-equal? (commission 0 88 50) (quote 500.0))
(check-equal? (commission 0 88 98) (quote 692.0))
(check-equal? (commission 0 88 99) (quote 696.0))
```

Test Analysis

All the Tennis-generated function tester programs provided as examples in this document can be executed by issuing the command: “racket [filename]”. Upon execution, each of these testers completes without notice of any failures, thus indicating that all problems used as examples are error free, as far as can be determined by boundary-value analysis, robust boundary-value analysis, and worst-case boundary-value analysis.