[SPOILERS] Practice Problems for Section 6 -- Tests

Subscribe for email updates. PINNED

No tags yet. + Add Tag

Pavel Lepin COMMUNITY TA · a month ago

```
#lang racket
;; Based on the provided test file for HW4 in proglang.

;; INSTRUCTIONS:
;; Put into the same folder as the file with your practice problem solutions.
;; Make sure it starts with:
;;     #lang racket
;;     (provide (all-defined-out))
(require "Put the name of the file with your solutions here")
(require rackunit)
(require rackunit/text-ui)

(define tests
  (test-suite
   "Model tests for section 6 practice problems"

   ;;; The Usual Suspects ;;;

   (check-equal? (btree-fold string-append "!"
                             (btree-node "foo" (btree-node "bar" (btree-leaf) (btree-l
eaf))
                                         (btree-node "baz" (btree-leaf) (btree-leaf)))
)
                 "!bar!foo!baz!" "btree-fold test #1")
   (check-equal? (btree-fold * 1 (btree-leaf)) 1 "btree-fold #2")
   (check-equal? (btree-fold (lambda (l v r) (- v l r)) 1
                             (btree-node 10 (btree-node 5 (btree-leaf) (btree-leaf))
                                         (btree-node 3 (btree-leaf) (btree-leaf))))
                 6 "btree-fold test #3")

   (check-equal? (btree-unfold
```

```
                 (lambda (x)
                   (if (zero? x)
                       #f
                       (cons x (cons (sub1 x) (sub1 x)))))
                 2)
                (btree-node 2 (btree-node 1 (btree-leaf) (btree-leaf))
                            (btree-node 1 (btree-leaf) (btree-leaf)))
                "btree-unfold test #1")
   (check-equal? (btree-unfold (lambda (x) #f) #f) (btree-leaf) "btree-unfold test #2"
)
   (check-equal? (btree-unfold
                 (lambda (x)
                   (if (zero? x)
                       #f
                       (cons x (cons (quotient x 2) (quotient x 3)))))
                 6)
                (btree-node 6 (btree-node 3 (btree-node 1 (btree-leaf) (btree-leaf))
                                            (btree-node 1 (btree-leaf) (btree-leaf)))
                            (btree-node 2 (btree-node 1 (btree-leaf) (btree-leaf))
                                          (btree-leaf)))
                "btree-unfold test #3")

   (check-equal? (gardener (btree-leaf)) (btree-leaf) "gardener test #1")
   (check-equal? (gardener
                 (btree-node 10
                             (btree-node "some string"
                                         (btree-node #f
                                                     (btree-node 1 (btree-leaf) (btre
e-leaf))
                                                     (btree-node (lambda (x) x)
                                                                 (btree-node "foobar"
                                                                             (btree-l
eaf)
                                                                             (btree-l
eaf))
                                                                 (btree-leaf)))
                                         (btree-node "hm..."
                                                     (btree-node 20 (btree-leaf) (btr
ee-leaf))
                                                     (btree-node #f (btree-leaf) (btr
ee-leaf))))
                             (btree-node #t (btree-leaf)
                                         (btree-node (list 1 2 3) (btree-leaf) (btree
-leaf)))))
                (btree-node 10
```

```
                            (btree-node "some string"
                                        (btree-leaf)
                                        (btree-node "hm..."
                                                    (btree-node 20 (btree-leaf) (btre
e-leaf))
                                                    (btree-leaf)))
                            (btree-node #t (btree-leaf)
                                        (btree-node (list 1 2 3) (btree-leaf) (btree-
leaf))))
                  "gardener test #2")
   (check-equal? (gardener
                  (btree-node #f
                              (btree-node "some string"
                                          (btree-node #f
                                                      (btree-node 1 (btree-leaf) (btre
e-leaf))
                                                      (btree-node (lambda (x) x)
                                                                  (btree-node "foobar"
                                                                              (btree-l
eaf)
                                                                              (btree-l
eaf))
                                                                  (btree-leaf)))
                                          (btree-node "hm..."
                                                      (btree-node 20 (btree-leaf) (btr
ee-leaf))
                                                      (btree-node #f (btree-leaf) (btr
ee-leaf))))
                              (btree-node #t (btree-leaf)
                                          (btree-node (list 1 2 3) (btree-leaf) (btree
-leaf)))))
                  (btree-leaf)
                  "gardener test #3")

   ;;; So Dynamic ;;;

   ;; Crazy Sum ;;
   (check-equal? (crazy-sum (list 10 * 6 / 5 - 3)) 9 "crazy-sum test #1")
   (check-equal? (crazy-sum (list 1 2 3 4 5 * 6)) 90 "crazy-sum test #2")
   (check-equal? (crazy-sum (list 6 - 3 6 9 / 6 + 5)) 3 "crazy-sum test #3")

   ;; Universal Fold ;;

   (check-equal? (universal-fold (lambda (x y) x) 0 null) 0 "universal-fold test #1")
   (check-equal? (universal-fold string-append "" (list "a" "b" "c")) "abc" "universal
```

```
-fold test #2")
   (check-equal? (universal-fold * 1 (btree-node 1 (btree-node 2 (btree-node 3 (btree-
leaf) (btree-leaf))
                                                                  (btree-node 4 (btree-le
af) (btree-leaf)))
                                                 (btree-node 5 (btree-leaf) (btree-lea
f)))) 120 "universal-fold test #3")
   (check-equal? (universal-fold (lambda (x y) y) #t (btree-leaf)) #t "universal-fold
test #4")

   (check-equal? (universal-sum null) 0 "universal-sum test #1")
   (check-equal? (universal-sum (list 10 21 32)) 63 "universal-sum test #2")
   (check-equal? (universal-sum (btree-node 1 (btree-node 2 (btree-node 3 (btree-leaf)
 (btree-leaf))
                                                          (btree-node 4 (btree-leaf) (
btree-leaf)))
                                                (btree-node 5 (btree-leaf) (btree-leaf))))
 15 "universal-sum test #3")
   (check-equal? (universal-sum (btree-leaf)) 0 "universal-sum test #4")

   ;; Stomp! ;;
   (check-equal? (flatten (list 1 2 (list (list 3 4) 5 (list (list 6) 7 8)) 9 (list 10
))) (list 1 2 3 4 5 6 7 8 9 10)
                 "flatten test #1")
   (check-equal? (flatten null) null "flatten test #2")
   (check-equal? (flatten (list (list (list "a" "b" "c")) (list "d" "e") "f")) (list "
a" "b" "c" "d" "e" "f")
                 "flatten test #3")
   (check-equal? (flatten (list (list null) (list "c" null "d") "e")) (list "c" "d" "e
")
                 "flatten test #4")

   ;;; Lambda Madness ;;;
   (check-equal? (simplify (mlet "x" (add (int 1) (int 2)) (var "x")))
                 (call (fun #f "x" (var "x")) (add (int 1) (int 2)))
                 "simplify test #1")
   (check-equal? (simplify (fun #f "x" (mlet "p" (apair (aunit) (var "x")) (apair (snd
 (var "p")) (fst (var "p"))))))
                 (fun #f "x" (call (fun #f "p" (fun #f "_f"
                                                       (call (call (var "_f") (call (var
"p") (fun #f "_x" (fun #f "_y" (var "_y")))))
                                                             (call (var "p") (fun #f "_x"
 (fun #f "_y" (var "_x")))))))
                                     (fun #f "_f" (call (call (var "_f") (aunit)) (var "
x")))))
```

```
                "simplify test #2")
  ))

(run-tests tests)
```

1 · flag

New post

To ensure a positive and productive discussion, please read our forum posting policies before posting.

B | I | Link | <code> | Pic | Math | Edit: Rich | Preview

Make this post anonymous to other students

Subscribe to this thread at the same time

Add post