CS5010 - Problem Set 08 - Test Results

pdp-pair-pankajt-wangzet

November 11, 2014

This test suite tests your implementation of Problem Set 08

1 File: pitchers.rkt

1.1 Test-Group: pitchers-after-moves (4 Points)

8/3/4

Common Definitions

```
(define PITCHERS-0 (list-to-pitchers '((1 1))))

(define PITCHERS-1 (list-to-pitchers '((8 8) (5 0) (3 0))))

(define PITCHERS-2 (list-to-pitchers '((8 3) (5 5) (3 0))))

(define PITCHERS-3 (list-to-pitchers '((8 0) (3 3) (5 5))))

(define PITCHERS-4
(list-to-pitchers '((12 4) (5 5) (3 3) (2 0) (4 0) (24 0) (0 0))))

(define MOVES-1 (list (make-move 1 2)))

(define MOVES-2 (list (make-move 1 2) (make-move 2 1)))

(define MOVES-3 (list (make-move 1 3)))

(define MOVES-4 (list (make-move 2 5) (make-move 3 6)))

(define MOVES-5
(list
 (make-move 1 6)
 (make-move 2 6)
 (make-move 3 6)
 (make-move 6 7)
 (make-move 6 5)
 (make-move 5 3)
 (make-move 6 1)))
```

1.1.1 Test (equality, 1/3 partial points)

pitchers-to-list/list-to-pitchers
Input:

```
(pitchers-to-list (list-to-pitchers '((8 8) (5 0) (3 0))))
```

Expected Output:

```
'((8 8) (5 0) (3 0))
```

Expected Output Value:

```
((8 8) (5 0) (3 0))
```

Correct

1.1.2 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-2 empty))
```

Expected Output:

```
'((8 3) (5 5) (3 0))
```

Expected Output Value:

```
((8 3) (5 5) (3 0))
```

Correct

1.1.3 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-1 MOVES-1))
```

Expected Output:

```
'((8 3) (5 5) (3 0))
```

Expected Output Value:

```
((8 3) (5 5) (3 0))
```

Correct

1.1.4 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-1 MOVES-2))
```

Expected Output:

```
'((8 8) (5 0) (3 0))
```

Expected Output Value:

```
((8 8) (5 0) (3 0))
```

Correct

1.1.5 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-3 MOVES-2))
```

Expected Output:

```
'((8 3) (3 0) (5 5))
```

Expected Output Value:

```
((8 3) (3 0) (5 5))
```

Correct

1.1.6 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-1 MOVES-3))
```

Expected Output:

```
'((8 5) (5 0) (3 3))
```

Expected Output Value:

```
((8 5) (5 0) (3 3))
```

Correct

1.1.7 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-3 MOVES-3))
```

Expected Output:

```
'((8 0) (5 5) (3 3))
```

Expected Output Value:

```
((8 0) (5 5) (3 3))
```

Wrong Output:

```
((8 0) (3 3) (5 5))
```

1.1.8 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-4 MOVES-4))
```

Expected Output:

```
'((12 4) (5 1) (3 0) (2 0) (4 4) (24 3) (0 0))
```

Expected Output Value:

```
((12 4) (5 1) (3 0) (2 0) (4 4) (24 3) (0 0))
```

Correct

1.1.9 Test (equality, 1/3 partial points)

Input:

```
(pitchers-to-list (pitchers-after-moves PITCHERS-4 MOVES-5))
```

Expected Output:

```
'((12 8) (5 0) (3 3) (2 0) (4 1) (24 0) (0 0))
```

Expected Output Value:

```
((12 8) (5 0) (3 3) (2 0) (4 1) (24 0) (0 0))
```

Correct

1.2 Test-Group: solution tests (5 Points)

5/5

1.2.1 Test (equality)

Input:

```
(solution '(2) 1)
```

Expected Output:

```
false
```

Expected Output Value:

```
#f
```

Correct

1.2.2 Test (equality)

Input:

```
(solution '(2) 2)
```

Expected Output:

```
empty
```

Expected Output Value:

```
()
```

Correct

1.2.3 Test (predicate, 1/2 partial points)

Input:

```
(solution '(8 5 3) 4)
```

Output should match:

```
(check-moves '(8 5 3) 4)
```

Correct

1.2.4 Test (predicate, 1/2 partial points)

Input:

```
(solution '(10 7 3) 5)
```

Output should match:

```
(check-moves '(10 7 3) 5)
```

Correct

1.2.5 Test (equality, 1/2 partial points)

Input:

```
(solution '(8 4 2) 3)
```

Expected Output:

```
false
```

Expected Output Value:

```
#f
```

Correct

1.2.6 Test (equality, 1/2 partial points)

Input:

```
(solution '(9 8 7) 4)
```

Expected Output:

```
false
```

Expected Output Value:

```
#f
```

Correct

1.2.7 Test (predicate, 1/2 partial points)

Input:

```
(solution '(8 5 3 120) 4)
```

Output should match:

```
(check-moves '(8 5 3 120) 4)
```

Correct

1.2.8 Test (predicate, 1/2 partial points)

Input:

```
(solution '(16 11 7) 9)
```

Output should match:

```
(check-moves '(16 11 7) 9)
```

Correct

1.2.9 Test (predicate, 1/2 partial points)

Input:

```
(solution '(8 5 3) 7)
```

Output should match:

```
(check-moves '(8 5 3) 7)
```

Correct

1.2.10 Test (predicate, 1 partial points)

Input:

```
(solution '(5 1 1 1 1) 2)
```

Output should match:

```
(check-moves '(5 1 1 1 1) 2)
```

Correct

2 File: river.rkt

2.1 Test-Group: pitchers-after-moves (6 Points)

5/2/6

Common Definitions

```
(define PITCHERS-0 (river:list-to-pitchers '((2 0) (5 0) (10 0))))


(define MOVES-0
(list
 (river:make-fill 2)
 (river:make-move 2 1)
 (river:make-move 2 3)
 (river:make-dump 1)
 (river:make-fill 2)
 (river:make-move 2 1)
 (river:make-move 2 3)
 (river:make-dump 1)))


(define PITCHERS-1 (river:list-to-pitchers '((13 0) (0 0) (5 0))))
```

```
(define MOVES-1
(list
 (river:make-fill 1)
 (river:make-move 1 2)
 (river:make-move 2 3)
 (river:make-move 1 3)
 (river:make-dump 3)
 (river:make-fill 2)
 (river:make-dump 1)
 (river:make-dump 2)))
```

2.1.1 Test (equality, 1/2 partial points)

pitchers-to-list/list-to-pitchers
Input:

```
(river:pitchers-to-list (river:list-to-pitchers '((8 8) (5 0) (3 0))))
```

Expected Output:

```
'((8 8) (5 0) (3 0))
```

Expected Output Value:

```
((8 8) (5 0) (3 0))
```

Correct

2.1.2 Test (equality, 1/2 partial points)

Input:

```
(river:pitchers-to-list (river:pitchers-after-moves PITCHERS-0 empty))
```

Expected Output:

```
'((2 0) (5 0) (10 0))
```

Expected Output Value:

```
((2 0) (5 0) (10 0))
```

Correct

2.1.3 Test (equality, 1/2 partial points)

Input:

```
(river:pitchers-to-list
(river:pitchers-after-moves PITCHERS-0 MOVES-0))
```

Expected Output:

```
'((2 0) (5 0) (10 6))
```

Expected Output Value:

```
((2 0) (5 0) (10 6))
```

Correct

2.1.4 Test (equality, 1/2 partial points)

Input:

```
(river:pitchers-to-list
(river:pitchers-after-moves PITCHERS-1 MOVES-1))
```

Expected Output:

```
'((13 0) (0 0) (5 0))
```

Expected Output Value:

```
((13 0) (0 0) (5 0))
```

Correct

2.1.5 Test (predicate, 1/2 partial points)

Input:

```
(river:solution '(7 3) 2)
```

Output should match:

```
(river:check-moves '(7 3) 2)
```

Wrong Output:

```
(#(struct:move 1 2)
#(struct:dump 1)
#(struct:move 2 1)
#(struct:fill 2)
#(struct:move 2 1)
#(struct:fill 2)
#(struct:move 2 1))
```

2.1.6 Test (predicate, 1/2 partial points)

Input:

```
(river:solution '(10 3 5) 1)
```

Output should match:

```
(river:check-moves '(10 3 5) 1)
```

Wrong Output:

```
(#(struct:move 1 2)
#(struct:move 1 3)
#(struct:dump 1)
#(struct:move 2 1)
#(struct:move 3 1)
#(struct:fill 2)
#(struct:move 2 1))
```

2.1.7 Test (equality, 1/2 partial points)

Input:

```
(river:solution '(8 4 2) 3)
```

Expected Output:

```
false
```

Expected Output Value:

```
#f
```

Correct

2.1.8 Test (predicate, 1/2 partial points)

Input:

```
(river:solution '(2 5 10) 6)
```

Output should match:

```
(river:check-moves '(2 5 10) 6)
```

Wrong Output:

```
(#(struct:move 1 2)
#(struct:fill 1)
#(struct:move 1 2)
#(struct:move 2 3)
#(struct:fill 1)
#(struct:move 1 3))
```

2.1.9 Test (predicate, 1/2 partial points)

Input:

```
(river:solution '(3 5) 4)
```

Output should match:

```
(river:check-moves '(3 5) 4)
```

Wrong Output:

```
(#(struct:move 1 2)
#(struct:fill 1)
#(struct:move 1 2)
#(struct:dump 2)
#(struct:move 1 2)
#(struct:fill 1)
#(struct:move 1 2))
```

2.1.10 Test (predicate, 1/2 partial points)

Input:

```
(river:solution '(11 7) 6)
```

Output should match:

```
(river:check-moves '(11 7) 6)
```

Wrong Output:

```
(#(struct:move 1 2)
#(struct:dump 1)
#(struct:move 2 1)
#(struct:fill 2)
#(struct:move 2 1)
#(struct:dump 1)
#(struct:move 2 1)
#(struct:fill 2)
#(struct:move 2 1)
#(struct:fill 2)
#(struct:move 2 1))
```

2.1.11 Test (predicate, 1/2 partial points)

Input:

```
(river:solution '(8 8 40 80 120 5 5) 4)
```

Output should match:

```
(river:check-moves '(8 8 40 80 120 5 5) 4)
```

Wrong Output:

```
(#(struct:move 1 2)
#(struct:move 2 6)
#(struct:move 2 3)
#(struct:move 6 1)
#(struct:move 3 4)
#(struct:move 1 2)
#(struct:move 4 1)
#(struct:move 2 3)
#(struct:move 1 5)
#(struct:move 3 4)
#(struct:move 5 6)
#(struct:move 4 5)
#(struct:move 6 7)
#(struct:fill 1)
#(struct:move 1 3)
#(struct:move 3 6)
#(struct:move 3 2)
#(struct:move 2 7)
#(struct:move 2 1)
#(struct:move 5 1)
#(struct:move 1 3)
#(struct:move 6 1)
#(struct:move 3 1)
#(struct:move 1 2)
#(struct:move 2 6)
#(struct:move 2 4)
#(struct:move 3 1)
#(struct:move 4 5)
#(struct:move 6 2)
#(struct:move 1 3)
#(struct:move 2 1)
#(struct:move 3 2)
#(struct:move 1 3)
#(struct:move 2 4)
#(struct:move 3 4)
#(struct:move 4 5)
#(struct:move 7 1)
#(struct:move 5 4)
#(struct:move 1 2)
#(struct:move 4 2)
#(struct:move 2 3)
```

```
#(struct:move 4 3)
#(struct:fill 1)
#(struct:move 1 3)
#(struct:move 3 4)
#(struct:move 4 2)
#(struct:move 2 3)
#(struct:move 3 6)
#(struct:move 3 1)
#(struct:move 4 1)
#(struct:move 1 5)
#(struct:move 4 1)
#(struct:move 1 3)
#(struct:move 3 7)
#(struct:move 3 2)
#(struct:move 4 1)
#(struct:move 2 1)
#(struct:move 1 5)
#(struct:move 5 2)
#(struct:move 2 3)
#(struct:move 5 2)
#(struct:move 3 1)
#(struct:move 6 2)
#(struct:move 1 4)
#(struct:move 2 3)
#(struct:move 3 6)
#(struct:move 6 1)
#(struct:move 3 1)
#(struct:move 1 6)
#(struct:move 1 2)
#(struct:move 3 5)
#(struct:move 4 3)
#(struct:move 2 1)
#(struct:move 3 1)
#(struct:move 6 4)
#(struct:move 1 2)
#(struct:move 2 6)
#(struct:move 2 3)
#(struct:move 3 1)
#(struct:move 4 2)
#(struct:move 1 4)
#(struct:move 2 1)
#(struct:move 4 2)
#(struct:move 1 3)
#(struct:move 2 5)
#(struct:move 5 1)
#(struct:move 1 3)
```

```
#(struct:move 3 4)
#(struct:move 4 2)
#(struct:move 2 3)
#(struct:move 4 1)
#(struct:move 3 4)
#(struct:move 1 2)
#(struct:move 5 1)
#(struct:move 2 3)
#(struct:move 1 2)
#(struct:move 3 5)
#(struct:move 2 3)
#(struct:move 4 1)
#(struct:move 3 4)
#(struct:move 1 3)
#(struct:move 4 1)
#(struct:move 3 5)
#(struct:move 6 2)
#(struct:move 1 2)
#(struct:move 2 3)
#(struct:move 5 2)
#(struct:move 2 3)
#(struct:move 3 4)
#(struct:move 5 2)
#(struct:move 4 1)
#(struct:move 1 2)
#(struct:move 1 3)
#(struct:move 2 3)
#(struct:move 4 1)
#(struct:move 3 4)
#(struct:move 4 2)
#(struct:move 2 3)
#(struct:move 4 1)
#(struct:move 1 2)
#(struct:move 4 1)
#(struct:move 7 3)
#(struct:move 1 5)
#(struct:move 2 1)
#(struct:move 3 4)
#(struct:move 1 3)
#(struct:move 3 6)
#(struct:move 3 7)
#(struct:move 4 2)
#(struct:move 2 3)
#(struct:move 4 1)
#(struct:move 3 1)
#(struct:move 1 4)
```

```
#(struct:move 3 2)
#(struct:move 4 5)
#(struct:move 5 3)
#(struct:move 2 3)
#(struct:move 6 1)
#(struct:move 3 2)
#(struct:move 1 4)
#(struct:move 3 4)
#(struct:move 2 4)
#(struct:move 4 5)
#(struct:move 5 1)
#(struct:move 1 3)
#(struct:move 3 6)
#(struct:move 3 2)
#(struct:move 5 3)
#(struct:move 2 1)
#(struct:move 3 2)
#(struct:move 1 4)
#(struct:move 2 5)
#(struct:move 3 1)
#(struct:move 4 2)
#(struct:move 1 4)
#(struct:move 2 1)
#(struct:move 1 7)
#(struct:move 1 3)
#(struct:move 4 1)
#(struct:move 5 2)
#(struct:move 6 1)
#(struct:move 1 3)
#(struct:move 2 3)
#(struct:move 3 4)
#(struct:move 4 1)
#(struct:move 1 3)
#(struct:move 4 1)
#(struct:move 1 3)
#(struct:move 3 5)
#(struct:move 4 1)
#(struct:move 5 2)
#(struct:move 2 1)
#(struct:move 1 3)
#(struct:move 5 2)
#(struct:move 2 4)
#(struct:move 3 4)
#(struct:move 5 2)
#(struct:move 6 1)
#(struct:move 2 3)
```

```
 #(struct:move 1 2)
 #(struct:move 3 1)
 #(struct:move 2 3)
 #(struct:move 1 6)
 #(struct:move 3 4)
 #(struct:move 4 1)
 #(struct:move 1 6))
```

3 Results

Successes: 23
Wrong Outputs: 7
Errors: 0
Achieved Points: 61/6
Total Points (rounded): 10/15