Dominoes Haskell Assignment

To test each function, I will give both true and false inputs to prove the correctness of the function in question.

**goesP :: Domino -> End -> Board -> Bool**

True and false scenarios for goesP on left side of board:

**\*Dominoes>** goesP (1,0) L [(1,1), (1,3), (3,3)]

True

**\*Dominoes>** goesP (0,0) L [(1,1), (1,3), (3,3)]

False

True and false scenarios for the right side of the board:

**\*Dominoes>** goesP (3,0) R [(1,1), (1,3), (3,3)]

True

**\*Dominoes>** goesP (0,0) R [(1,1), (1,3), (3,3)]

False

Test for trying to place a domino in an empty board:

**\*Dominoes>** goesP (1,0) L []

True

**\*Dominoes>** goesP (1,0) R []

True

**knockingP :: Hand -> Board -> Bool**

Test if player is “knocking” or not:

**\*Dominoes>** knockingP [(0,0), (1,1), (2,2)] [(3,3), (3,5), (5,5)]

True

**\*Dominoes>** knockingP [(0,3), (1,1), (2,2)] [(3,3), (3,5), (5,5)]

False

Test for empty hand:

**\*Dominoes>** knockingP [] [(1,0)]

True

Test for empty board:

**\*Dominoes>** knockingP [(1,0)] []

False

**playedP :: Domino -> Board -> Bool**

Test for domino that has been played:

**\*Dominoes>** playedP (0,2) [(0,0), (0,2), (2,2)]

True

Test for domino that has not been played:

**\*Dominoes>** playedP (3,3) [(0,0), (0,2), (2,2)]

False

Test for empty board:

**\*Dominoes>** playedP (1,0) []

False

**possPlays :: Hand -> Board -> ([Domino], [Domino])**

Test for dominoes on right and left:

**\*Dominoes>** possPlays [(0,1), (1,2), (1,3)] [(0,0), (0,2), (2,3)]

([(0,1)],[(1,3)])

Test for just dominoes on the left:

**\*Dominoes>** possPlays [(0,1), (1,2), (1,3)] [(0,0), (0,6), (6,6)]

([(0,1)],[])

Test for dominoes on the right:

**\*Dominoes>** possPlays [(0,1), (1,2), (1,3)] [(6,6), (6,2), (2,3)]

([],[(1,3)])

Test for no dominoes playable:

**\*Dominoes>** possPlays [(0,0), (1,1), (2,2)] [(3,3), (3,5), (5,5)]

([],[])

Test for empty hand:

**\*Dominoes>** possPlays [] [(1,0)]

([],[])

Test for empty board:

**\*Dominoes>** possPlays [(1,0)] []

([(1,0)],[(1,0)])

**playDom :: Domino -> End -> Board -> Maybe Board**

I started by placing the domino in an empty board:

**\*Dominoes>** playDom (1,1) L []

Just [(1,1)]

I placed a domino on the left of the board twice here, once it had to be flipped and the other time it fit in without being flipped:

**\*Dominoes>** playDom (1,0) L [(0,2)]

Just [(1,0),(0,2)]

**\*Dominoes>** playDom (0,1) L [(0,2)]

Just [(1,0),(0,2)]

Then I did the same with the right side of the board:

**\*Dominoes>** playDom (2,1) R [(0,2)]

Just [(0,2),(2,1)]

**\*Dominoes>** playDom (1,2) R [(0,2)]

Just [(0,2),(2,1)]

Here I tried to place the domino in a board it does not fit in:

**\*Dominoes>** playDom (1,1) R [(0,2)]

Nothing

And again, on the left:

**\*Dominoes>** playDom (1,1) L [(0,2)]

Nothing

The same on a board of length more than one:

**\*Dominoes>** playDom (1,1) L [(0,2),(2,3),(3,6)]

Nothing

**\*Dominoes>** playDom (1,1) R [(0,2),(2,3),(3,6)]

Nothing

**scoreBoard :: Board -> Int**

There are lots of different ways to test scoreBoard, so I will start by explaining what each of the tests does. First, I will test for a board with a score of 0:

**\*Dominoes>** scoreBoard [(0,0),(0,2),(2,2)]

0

For a double on the right side (3):

**\*Dominoes>** scoreBoard [(0,1),(1,3),(3,3)]

2

Double on right side (5):

**\*Dominoes>** scoreBoard [(0,1),(1,5),(5,5)]

2

For a double on the left side (3):

**\*Dominoes>** scoreBoard [(3,3),(3,1),(1,0)]

2

For a double on both sides:

**\*Dominoes>** scoreBoard [(3,3),(3,6),(6,6)]

6

Here I have tested for a (3) score on the right side:

**\*Dominoes>** scoreBoard [(0,2),(2,5),(5,3)]

1

And for a (5) score on the right side:

**\*Dominoes>** scoreBoard [(0,2),(2,3),(3,5)]

1

And for a (5) score on the left side:

**\*Dominoes>** scoreBoard [(5,3),(3,0),(0,0)]

1

A (3) score on the left side:

**\*Dominoes>** scoreBoard [(3,5),(5,0),(0,0)]

1

And finally, I have tested for a 3 and 5 score:

**\*Dominoes>** scoreBoard [(6,6),(6,0),(0,3)]

8

**scoreN :: Board -> Int -> ([Domino],[Domino])**

Test with n = 0 on an empty board to see all the dominoes that could be played without returning a score:

**\*Dominoes>** scoreN [] 0

([(5,6),(4,4),(3,5),(3,4),(2,6),(2,5),(2,2),(1,6),(1,3),(1,1),(0,4),(0,2),(0,1),(0,0)],[(5,6),(4,4),(3,5),(3,4),(2,6),(2,5),(2,2),(1,6),(1,3),(1,1),(0,4),(0,2),(0,1),(0,0)])

Test with n = 1 on an empty board:

**\*Dominoes>** scoreN [] 1

([(2,3),(1,4),(1,2),(0,5),(0,3)],[(2,3),(1,4),(1,2),(0,5),(0,3)])

Test with n = 100 on an empty board to show that no dominoes can be played to obtain too high a score:

**\*Dominoes>** scoreN [] 100

([],[])

Test with the given example in the assignment brief:

**\*Dominoes>** scoreN [(5,2),(2,4),(4,4),(4,0)] 2

([(5,6),(5,5)],[(0,5),(0,1)])

Test with n = 100:

**\*Dominoes>** scoreN [(5,2),(2,4),(4,4),(4,0)] 100

([],[])

Test with random board and n = 2:

**\*Dominoes>** scoreN [(3,3),(3,2),(2,5),(5,6)] 2

([(3,4),(0,3)],[(4,6),(0,6)])

Test with same board but n = 3:

**\*Dominoes>** scoreN [(3,3),(3,2),(2,5),(5,6)] 3

([],[(3,6)])

Test with new board and n = 0:

**\*Dominoes>** scoreN [(5,5),(5,4),(4,2),(2,3)] 0

([(4,5),(1,5)],[(3,6),(3,3)])

Test with same board but n = 3:

**\*Dominoes>** scoreN [(5,5),(5,4),(4,2),(2,3)] 3

([(5,6)],[(1,3)])

Test with same board but n = 100:

**\*Dominoes>** scoreN [(5,5),(5,4),(4,2),(2,3)] 100

([],[])