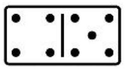
University of Sheffield, Department of Computer Science

COM2001: Advanced Programming Techniques
Semester 1: Functional Programming
Assessment 2017-18
Assignment 2: Domino Games

This exercise counts for 12.5% of the assessment for the COM2001 module

1. Introduction

In this assignment you will write code which will play a round of 5s-and-3s dominoes between two computer players. The rules are

- Each player starts with a hand of 9 dominoes. The remaining 10 dominoes take no part in this round – they are 'sleeping'.
- Players add a domino to the board ('dropping' a domino) in turn and accumulate the 5s-and-3s scores of their 'drops'.
- The player who drops the first domino gets the 5s-and-3s score of its total spots (e.g. if the first drop is  the player who dropped it scores 3.
- Play continues until neither player can play (either because s/he has run out of dominos or is knocking).

2. Computer Players

Define a Haskell datatype to represent a dominoes player. A **DomsPlayer** will be a function which takes a **Hand** and a **Board** and returns a **Domino** to play and the **End** to play it at, i.e. each time it is called it makes a single move.

There is no need to check whether the player is knocking: the code below will only call the player if there is at least one domino in its hand which can be dropped.

In this assignment you need not devote much effort to developing clever **DomsPlayers**: that is the topic of assignment 3. You will, however, need at least one to test your game-playing mechanism. You could have

simplePlayer :: DomsPlayer -- simplePlayer plays the first domino in its hand which will go.

hsdPlayer :: DomsPlayer – hsdPlayer always plays the highest scoring domino in its hand.

3. The Deal

Write **shuffleDoms,** which takes a random number generator (explained in the lectures) and returns a list of all the dominoes in random order. You will need to import **System.Random** and to define the list of all Dominoes as a constant.

4. Playing the round

Write a function **playDomsRound** which takes 2 **DomsPlayer**s and an **Int** which is a seed for the random number generator , uses **shuffleDoms** to define their initial **Hands** and calls each player in turn, updating the **Board, Hands** and scores**,** returning the final score at the end of the round.

5. Design

You will find that, in addition to the functions above, you will need to write several other 'helper' functions. **You should think through the organisation of your code before you start to write it.** You will need to clarify the data structures, the role of each function and its type declaration (i.e. its arguments and what it returns). You might do this in a diagram.

6. Mark Scheme

	credit
Design	20
Data Structures	15
Doms Players	15
shuffleDoms	15
playDomsRound	35

Assessment Criteria*: In function definitions, 60% of the credit is for coding, 20% for testing and 20% for documentation. A function which is working but poorly coded will be awarded about ½ of the credit for coding (i.e. 30% of the total credit for the function). The remaining coding credit is for good use of Haskell and the quality of the code.*

Late hand-in penalties: see

http://www.dcs.shef.ac.uk/intranet/teaching/public/assessment/latehandin.html

Plagiarism penalties: see

http://www.dcs.shef.ac.uk/intranet/teaching/public/assessment/plagiarism.html

7. What to hand in

Hand in 3 documents, in a single ZIP archive:

1. Your design,
2. Your commented code as a .hs ,file, ready to run,
3. Your test results.

6. How to hand in

Hand in by MOLE

DEADLINE: Midnight Monday 20th November (week 9)