Functional programming

Final assignment

deadline 04-10-2021 8:00

Rules

The assignment should be done individually. You are allowed to discuss the algorithm for solving this with your fellow students, as it can be quite tricky, but the implementations in Haskell and Java must be done individually. You have to **hand in the assignment by Monday 04-10-2021 before 8:00.** Handing in the assignment can be done by either sending me the deliverables by email, or uploading your deliverables to your github repository and providing me the link.

I expect you to deliver the following:

- A Haskell script file containing the solution. As discussed in the lectures, I expect all top-level functions to have proper type annotations! You are of course allowed to define local functions using *where* without type annotations.
- A Java implementation of the same algorithm. It is not necessary to provide unit tests for your code, but it is advisable to test your code to such a level that you are confident your solution works.
- A short report discussing the following:
  - A description of the algorithm you used. Without reading the code, I should be able to understand how you solve the problem.
  - How to run both the Haskell and Java implementations of the program
  - A comparison of the implementations: think about things like clarity of the implementation, the ease of implementation, the amount of ‘boilerplate code’, and how easy it is to modify and test (parts of) the algorithm.
- The report should be no longer than 3 pages A4

The assignment: domino effect

A standard set of Double Six dominoes contains 28 pieces (called bones) each displaying two numbers from 0 (blank) to 6 using dice-like pips. The 28 bones, which are unique, consist of the following combinations of pips:

Bone #	Pips	Bone #	Pips	Bone #	Pips	Bone #	Pips
1	0 | 0	8	1 | 1	15	2 | 3	22	3 | 6
2	0 | 1	9	1 | 2	16	2 | 4	23	4 | 4
3	0 | 2	10	1 | 3	17	2 | 5	24	4 | 5
4	0 | 3	11	1 | 4	18	2 | 6	25	4 | 6
5	0 | 4	12	1 | 5	19	3 | 3	26	5 | 5
6	0 | 5	13	1 | 6	20	3 | 4	27	5 | 6
7	0 | 6	14	2 | 2	21	3 | 5	28	6 | 6

All the Double Six dominoes in a set can he laid out to display a 7 x 8 grid of pips. Each layout corresponds at least one "map" of the dominoes. A map consists of an identical 7 x 8 grid with the appropriate bone numbers substituted for the pip numbers appearing on that bone. An example of a 7 x 8 grid display of pips and a corresponding map of bone numbers is shown below.

```
7 x 8 grid of pips                 map of bone numbers

6  6  2  6  5  2  4  1             28 28 14  7 17 17 11 11

1  3  2  0  1  0  3  4             10 10 14  7  2  2 21 23

1  3  2  4  6  6  5  4             8   4 16 25 25 13 21 23

1  0  4  3  2  1  1  2             8   4 16 15 15 13  9  9

5  1  3  6  0  4  5  5             12 12 22 22  5  5 26 26

5  5  4  0  2  6  0  3             27 24 24  3  3 18  1 19

6  0  5  3  4  2  0  3             27  6  6 20 20 18  1 19
```

Input and Output

You can handle input any way you like, and you can prepare some sample inputs in your script (for example those above. The program should print both the input, and the results. See below for an example.

Sample Solution

```
Input:
    5    4    3    6    5    3    4    6

    0    6    0    1    2    3    1    1

    3    2    6    5    0    4    2    0

    5    3    6    2    3    2    0    6

    4    0    4    1    0    0    4    1

    5    2    2    4    4    1    6    5

    5    5    3    6    1    2    3    1

Result(s):
    6   20   20   27   27   19   25   25

    6   18    2    2    3   19    8    8

   21   18   28   17    3   16   16    7

   21    4   28   17   15   15    5    7

   24    4   11   11    1    1    5   12

   24   14   14   23   23   13   13   12

   26   26   22   22    9    9   10   10
```

```
Input:
   4   2   5   2   6   3   5   4
   5   0   4   3   1   4   1   1
   1   2   3   0   2   2   2   2
   1   4   0   1   3   5   6   5
   4   0   6   0   3   6   6   5
   4   0   1   6   4   0   3   0
   6   5   3   6   2   1   5   3

Maps resulting from layout #2 are:
  16  16  24  18  18  20  12  11
   6   6  24  10  10  20  12  11
   8  15  15   3   3  17  14  14
   8   5   5   2  19  17  28  26
  23   1  13   2  19   7  28  26
  23   1  13  25  25   7  21   4
  27  27  22  22   9   9  21   4

  16  16  24  18  18  20  12  11
   6   6  24  10  10  20  12  11
   8  15  15   3   3  17  14  14
   8   5   5   2  19  17  28  26
  23   1  13   2  19   7  28  26
  23   1  13  25  25   7   4   4
  27  27  22  22   9   9  21  21
```