The University of Hong Kong

COMP3258: Functional Programming

Assignment 1 (Functions and Recursion)

Deadline: 23:55, Oct 08, 2020 (HKT)

Problem 1. (10 pts.) The function `rem` returns reminder of integer division of the arguments [http://zvon.org/other/haskell/Outputprelude/rem_f.html]. And the function `quot`, which divides the first argument by the second one discarding remainder [http://zvon.org/other/haskell/Outputprelude/quot_f.html].

You get n boxes of moon cakes and m friends (n larger than m). You want to make a Mid-Autumn Festival present with moon cakes to each friend. You should do this in the fairest (that is, most equal) manner and the friend who gets the most number should differ from the friend who gets the least number as little as possible.

Your task is to complete the function `solve :: Int -> Int -> [Int]`, where the output Integer list represent the i-th friend's number of moon cakes.

```
*Main> solve 12 3
[4, 4, 4]
*Main> solve 15 4
[3, 4, 4, 4]
*Main> solve 18 7
[2, 2, 2, 3, 3, 3, 3]
```

Problem 2. (10 pts.) Please write a `recursive` function `ismember` which, given a string and a list of strings, returns a Boolean indicating whether the string is in the list or not.

Example:

```
*Main> ismember "a" ["a","b","c"]
True
*Main> ismember "a" []
False
```

```
*Main> ismember "" [""]
True
```

Problem 3. (10 pts.) Roma (a popular Russian name that means 'Roman') loves the Little Lvov Elephant's lucky numbers.

Let us remind you that lucky numbers are positive integers whose decimal representation only contains lucky digits 4 and 7. For example, numbers 47, 744, 4 are lucky and 5, 17, 467 are not.

Roma got a list of positive integers. He wonders, how many of those integers have not more than k lucky digits? Help him, write the function `checkHappyDigits :: [Int] -> Int -> Int` that solves the problem.

Example:

```
*Main> checkHappyDigits [1,2,4] 4
3
*Main> checkHappyDigits [447,44,77] 2
2
```

Problem 4. (15 pts.) A binary tree is a tree which is an empty or a node with up to two subtree whose are binary tree as well.

A binary search tree is a binary tree if all the nodes are assigned a value and for any subtree, these rules are satisfied:

1. All the values in left subtree(if exists) are smaller than the value of the root.
2. All the values in right subtree(if exists) are greater than the value of the root.

Given `n` nodes, each having a unique value from `[1, N]`, your task is to implement function `numBST :: Int -> Int`, which compute the number of different binary search tree that can be constructed using all of them.

Since the answer might be too large, your answer should modulo $(10^9 + 7)$.

Expected running results:

```
*Main> numBST 1
1
*Main> numBST 3
5
*Main> numBST 4
14
```

```
*Main> numBST 100
558488487
```

Explanation For `n=3`, all the trees are:

```
1          1          2          3         3
 \          \        / \        /         /
  2          3      1   3      1         2
   \        /                   \       /
    3      2                     2     1
```

Problem 5. (15 pts.) Tom found a piece of paper with a coordinate system written on it. There are n distinct squares drawn in this coordinate system. The squares are numbered from 1 to n. The opposite corners of the i-th square are (0, 0), (ai, ai).

Tom wants to find such integer point (with integer coordinates) of the plane, that belongs to exactly k drawn squares. A point belongs to a square, if the point is located either inside the square, or on its boundary.

Help Tom find a point that would meet the described limits.

Write a function `pointIn :: [Int] -> Int -> [Int]` whose arguments are the integers list [a1,a2,...,an] and k (n larger than k). And then return the coordinates of the point that belongs to exactly k squares. If there are multiple answers, you are allowed to print any of them. If there is no answer, print [-1].

Expected running results:

```
*Main>  pointIn [5,1,3,4] 3
[2,1]
```

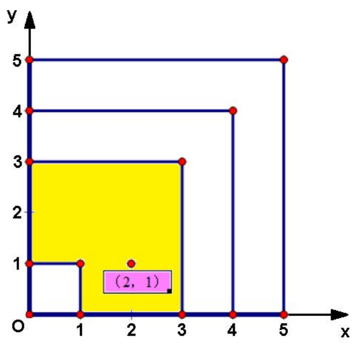

Figure 1 : example test case one

```
*Main>  pointIn [2,4,1] 1
[4,0]
```

Problem 6. (15 pts.) A list is defined to be twin paired if its even-valued elements (if any) are in ascending order and its odd-valued elements (if any) are in descending order.The list `{-6, 12, 5, 24, 3, 1}` is twin paired because the even-valued elements `(-6, 12, 24)` are in ascending order and the odd-valued elements `(5, 3, 1)` are in descending order. However, the list`{1, 2, 3}` is not twin paired because the odd numbers are not in descending order. Write a function named `isTwinPaired :: [Int] -> Bool` that returns `True` if its list argument is twin paired, otherwise it returns `False`.

Expected running results:

```
*Main> isTwinPaired [2,4,32]
True
*Main> isTwinPaired [2,2,2,1,1,1]
True
*Main> isTwinPaired [1,19,23]
False
*Main> isTwinPaired [3,2,1]
True
*Main> isTwinPaired [20,22,24,27,25]
True
```

Problem 7. (20 pts.) You are playing chess. Now you have a chessboard with size of $N \times N$ ($8 \leq N < 15$). You want to place N knights on the chessboard, and any pair of knight should not be conflicted. Two knights are considered conflicted if

1. They are lying on same row or column or a line drawn diagonally
2. A knight is lying on a "L-shape" location of the other knight. It means a knight is (2 rows, 1 column) or (1 row, 2 columns) away from the other knight.

See the picture below, if "k" represents a knight, then who are placed on squares marked with hyphens '-' are conflicted with the knight and who are placed on squares marked by '0' are safe.

```
- 0 0 - 0 0 -
0 - - - - - 0
0 - - - - - 0
- - - k - - -
0 - - - - - 0
0 - - - - - 0
- 0 0 - 0 0 -
```

Your task is to implement the function `chess :: Int -> Int`, which computes the number of ways to place N knights on an $N \times N$ chessboard such that none of knights are conflicted with each other. Ignore the fact that some of these arrangements are reflections and rotations of each other: all of them count as unique postions.

Expected running results:

```
*Main> chess 10
4
-- explanation --
-- there are 4 possible combinations:
-- (10,8), (9,5), (8,2), (7,10), (6,7), (5,4), (4,1), (3,9), (2,6), (1,3)
-- (10,7), (9,3), (8,10), (7,6), (6,2), (5,9), (4,5), (3,1), (2,8), (1,4)
-- (10,4), (9,8), (8,1), (7,5), (6,9), (5,2), (4,6), (3,10), (2,3), (1,7)
-- (10,3), (9,6), (8,9), (7,1), (6,4), (5,7), (4,10), (3,2), (2,5), (1,8)
```

Code style and submission (5 pts.)

All functions should be implemented in a single Haskell file, named as A1_XXX.hs, with XXX replaced by your UID. Your code should be well-written (e.g. proper indentation, names, and type annotations) and documented. Please submit your solution on Moodle before the deadline.

Notice: there are cases that students cannot upload a Haskell file on Moodle. Then please compress it into A1_XXX.zip, which contains only one file A1_XXX.hs.

Plagiarism

Please do this assignment on your own; if, for a small part of an exercise, you use something from the Internet or were advised by your classmate, please mark and attribute the source in a comment. Do not use publicly accessible code sharing websites for your assignment to avoid being suspected of plagiarism.