THE UNIVERSITY OF HONG KONG

COMP3258: FUNCTIONAL PROGRAMMING

Assignment 1 (Functions and Recursion)

Deadline: 23:55, Oct 11, 2019 (HKT)

Problem 1. (10 pts.) To produce sequences of random numbers, Pseudo Random Number Generator(PRNG)[https://en.wikipedia.org/wiki/Pseudorandom_number_generator] is used.

The most common algorithm is Linear Congruential Generator. The generator is defined by the recurrence relation: $X_n = (aX_{n-1} + b) \mod c$.

Here, multiplier a, increment b, modulus c and the seed or start value X_0 are given.

Your task is to complete the function `randGen idx x0 a b c`, where `idx` is the number of generated number in our algorithm.

```
*Main> randGen 1 0 1 1 10
1
*Main> randGen 5 0 1 1 10
5
*Main> randGen 10 0 1 1 10
0
*Main> randGen 2 3 4 3 50
13
```

Problem 2. (10 pts.) You get a sentence which contains many words. However, these words are separated by at least one space `' '`.

Please write a function `split :: String -> [String]`, which takes a string representing a sentence and returns a list of string representing the word list.

Example:

```
*Main> split "hello world"
["hello", "world"]
```

```
*Main> split " hello  world   again "
["hello", "world", "again"]
*Main> split " "
[]
```

Problem 3. (10 pts.) Sorting is really handy in Haskell. Run `sort [1,4,3,2]` and you can get `[1,2,3,4]` immediately. Let's make some interesting permutation from it!

Implement `wave :: [Int] -> [Int]`, which takes a list whose length is an odd number and returns a list that:

- The $n/2$ element is the largest one.
- The $n/2 - 1$ element is the second largest one.
- The $n/2 + 1$ element is the third largest one.
- The $n/2 - 2$ element is the fourth largest one.
- ...

Notice

- The return list should have the same length as the input, and contain all the elements in the input list.
- Duplication is allowed.

Expected running results:

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

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.

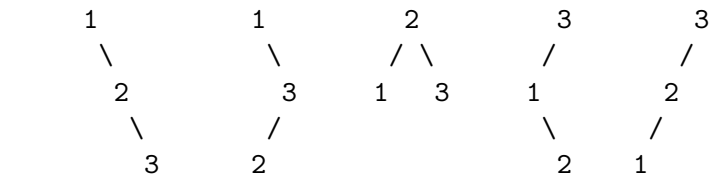
Given `n` nodes, each having a unique value from `[1, N]`, your task is to implement function `numBST :: Int -> Int`, which compute the number of different binary search tree that can be constructed using all of them.

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:



Problem 5. (15 pts.) A list is considered as a Martian list if the occurrence of 1s is greater than the occurrence of 2s.

For example, `[12, 23]` is not a Martian list, because the occurrence of 1s is 1 (exists in number 12) while the occurrence of 2s is 2 (exists in number 12 and 23).

Write a function named `isMartian` that returns True if its list argument is a Martian list, otherwise it returns False.

Expected running results:

```
*Main> isMartian [1,3]
True
*Main> isMartian [1,2,3]
False
*Main> isMartian [11,2]
True
*Main> isMartian [1,12]
```

```
True
*Main> isMartian [12]
False
```

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
```

```
O - - - - - O
- O O - O O -
```

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.