Programming Languages - Homework 2

Write Haskell functions for the following problems.

1. Finding Primes (30pts)

Print prime numbers using given inputs (N and M). You have to find M prime numbers which are bigger than or equal to N.

- Input : initial number ‘N’ and the number of primes ‘M’
- Output : a list of primes
- Report error if input is not correct.

```
e.g. ghci> FindingPrimes 2 5
     2 3 5 7 11
     ghci> FindingPrimes 6 6
     7 11 13 17 19 23
```

2. Flip Coins (30pts)

TA has a **stack** of coins. She has a strange hobby, which is to make all coins’ face same direction. Without scattering coins, she is supposed to make all coins head up by flipping the stack repeatedly. Let’s see following example, where H means Head and T means Tail.

```
flip "HTTHTHH" 5
"HTHHTHH"
flip "HTHHTHH" 4
"TTHTTHH"
```

First, flipping 5 coins changes “HTTHT” into “HTHHT” (=not “THTTH”) because the order is reversed and each coins’ face is flipped. In the same manner, flipping 4 coins makes “HTHH” become “TTHT”. Make a function that lists the indices of flipping to make all coins face same way as following described rule.

- Input : A string will consist of more than 2 coins and each coin will have a character either ‘H’ or ‘T’. The top coin on a stack appears first on the string, the bottom coin appears last.
- Output : A list of digits which are indices of flip (the index from the top, 1~n). If all coins face **Head**, put 0 which means the flipping is end.
- Report error if input is not correct.

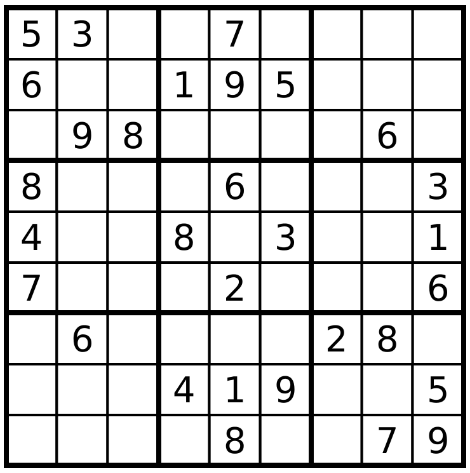
```
e.g. ghci> flipCoin "HT"
     [1,2,0]
     ghci> flipCoin "HTTH"
     [1,3,0]
     ghci> flipCoin "THTHTH"
     [5,3,1,2,4,0]
```

3. Squiggly Sudoku (Jigsaw Sudoku) [40pt]

TA is interested in solving sudoku puzzles. Becoming skillful to classic sudoku, he wanted to defy squiggly sudoku. But he couldn't clear any of them even though it was the easiest version. Thus he asked you to make squiggly sudoku solver and you should make the program.

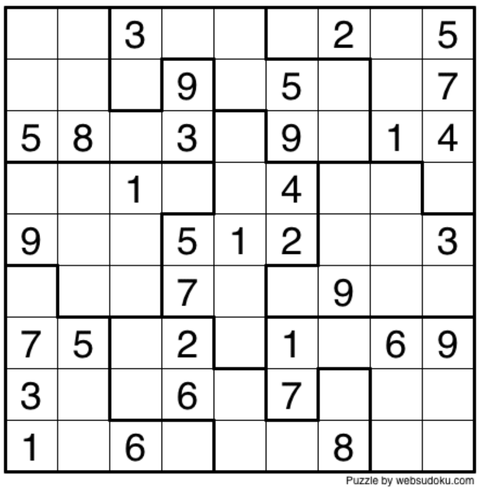
First, rules of the basic sudoku are below.

- fill a 9*9 grid with digits.
- each column, each row, and each of the nine 3*3 sub-grids have to contain all of the digits from 1 to 9.



Next, rules of squiggly sudoku are below.

- fill a 9*9 grid with digits.
- each column, each row, and each of bounded block have to contain all of the digits from 1 to 9.



Now, there are two given lists; one is a list which contains initial values, the other is a list of blocks and each cell in same block has same values. Make a function that returns a right solution of the input case.

```
 e.g. ghci> let valList =
[0,0,3,0,0,0,2,0,5,
0,0,0,9,0,5,0,0,7,
5,8,0,3,0,9,0,1,4,
0,0,1,0,0,4,0,0,0,
9,0,0,5,1,2,0,0,3,
0,0,0,7,0,0,9,0,0,
7,5,0,2,0,1,0,6,9,
3,0,0,6,0,7,0,0,0,
1,0,6,0,0,0,8,0,0]

ghci> let blkList =
[1,1,2,2,2,3,3,3,3,
1,1,2,1,2,2,2,3,3,
1,1,1,1,5,2,2,3,3,
4,4,4,4,5,5,6,6,3,
4,4,4,5,5,5,6,6,6,
7,4,4,5,5,6,6,6,6,
7,7,8,8,5,9,9,9,9,
7,7,8,8,8,9,8,9,9,
7,7,7,7,8,8,8,9,9]

ghci> squigglySudoku valList blkList

[#81 digits list of solution]
```

Due: **June 16 (Fri) 11:59 pm**

– Submit the code to GitLab.

Optionally zip the source code (ONLY .hs files) and submit it to the portal.