Homework assignment #1

Please submit a `.zip` archive containing a single file `homework1.hs` with your solutions to the problems below. Moodle blocks file submissions with `.hs` extension.

Good luck! 🍀

Acknowledgement: The problems of this homework assignment were suggested by Jason BILLARD.

Problem A: The Collatz Conjecture 🤔

The Collatz sequence, also known as the Collatz conjecture or the `3n+1` conjecture, is a mathematical sequence defined by a simple rule. Starting with any positive integer, if it's even, divide it by 2; if it's odd, multiply it by 3 and add 1. Repeating this process generates a sequence of numbers, and the conjecture suggests that, regardless of the starting integer, the sequence will always eventually reach the number 1.

(1) Create a Haskell function called `collatzStep :: Int -> Int` that takes an integer `n` as input and returns the result of one step of the Collatz sequence. If `n` is even, the function should return `n/2`, and if `n` is odd, it should return `3*n+1`.

```
Input: 1
Output: 4   -- Explanation: 1 is odd, so (1 * 3) + 1 = 4

Input: 6
Output: 3   -- Explanation: 6 is even, so 6 / 2 = 3
```

(2) Write a Haskell function named `countCollatzSteps :: Int -> Int` that takes an integer `n` as input and returns the number of steps required to reach `1` in the Collatz sequence, starting from `n`.

```
Input: 1
Output: 0
-- Starting at 1, it takes 0 steps to reach 1 (already there).

Input: 6
Output: 8
-- 6 -> 3 -> 10 -> 5 -> 16 -> 8 -> 4 -> 2 -> 1
-- Starting at 6, it takes 8 steps to reach 1.
```

(3) Generate an infinite list in Haskell called `stepsToOne :: [(Int,Int)]` that comprises pairs of integers, with the first element representing a sequentially ordered set of natural numbers, and the second element denoting the number of steps taken in the Collatz sequence for the corresponding number in the first element to reach the value `1`.

```
ghci> take 10 stepsToOne
[(1,0),(2,1),(3,7),(4,2),(5,5),(6,8),(7,16),(8,3),(9,19),(10,6)]
```

Problem B: Wake up! ⏰

Everyday, Tom's alarm clock goes off at irregular times, but it always rings at `00:00`. Tom needs to determine the earliest moment he can go to bed, after `00:00` to get an uninterrupted night's sleep for a duration of at least `n` minutes. Tom managed to find a way to acquire a list containing the specific times when the alarm will ring.

Help Tom determine the earliest time he can go to bed, given an ordered list of times when the alarm rings, and the number `n` of minutes he needs to sleep to wake up refreshed.

The input provided will always allow for an interval of sleep `n` and does not overlap to the following day.

```
type Time = (Int, Int)

goToBed :: [Time] -> Int -> Time
goToBed = undefined

testB1 = goToBed [(0, 00), (1,00), (3,13), (5,00), (7,43),
(15,05)] 140 == (5, 0)
testB2 = goToBed [(0, 00), (1,00), (3,13), (5,00), (7,43),
(15,05)] 240 == (7, 43)
testB3 = goToBed [(0, 00), (0, 55)] 60 == (0, 55)
testB4 = goToBed [(0, 00), (0, 55)] 45 == (0, 0)
```

Problem C: Tower of Hanoi 👓

The Tower of Hanoi is a classic mathematical puzzle featuring **three rods** and `n` different-sized disks. The challenge is to move a complete stack of disks from one rod to another while adhering to the rule that larger disks can never be placed on top of smaller ones.

Write a function `hanoi :: Int -> Int` that computes the number of steps needed to move the whole stack of `n` discs from one rod to another.

Place a comment in the code explaning your strategy!

```
{-
Strategy: ??
-}

hanoi :: Int -> Int
hanoi = undefined

testC1 = hanoi 3 == 7
testC2 = hanoi 14 == 16383
```

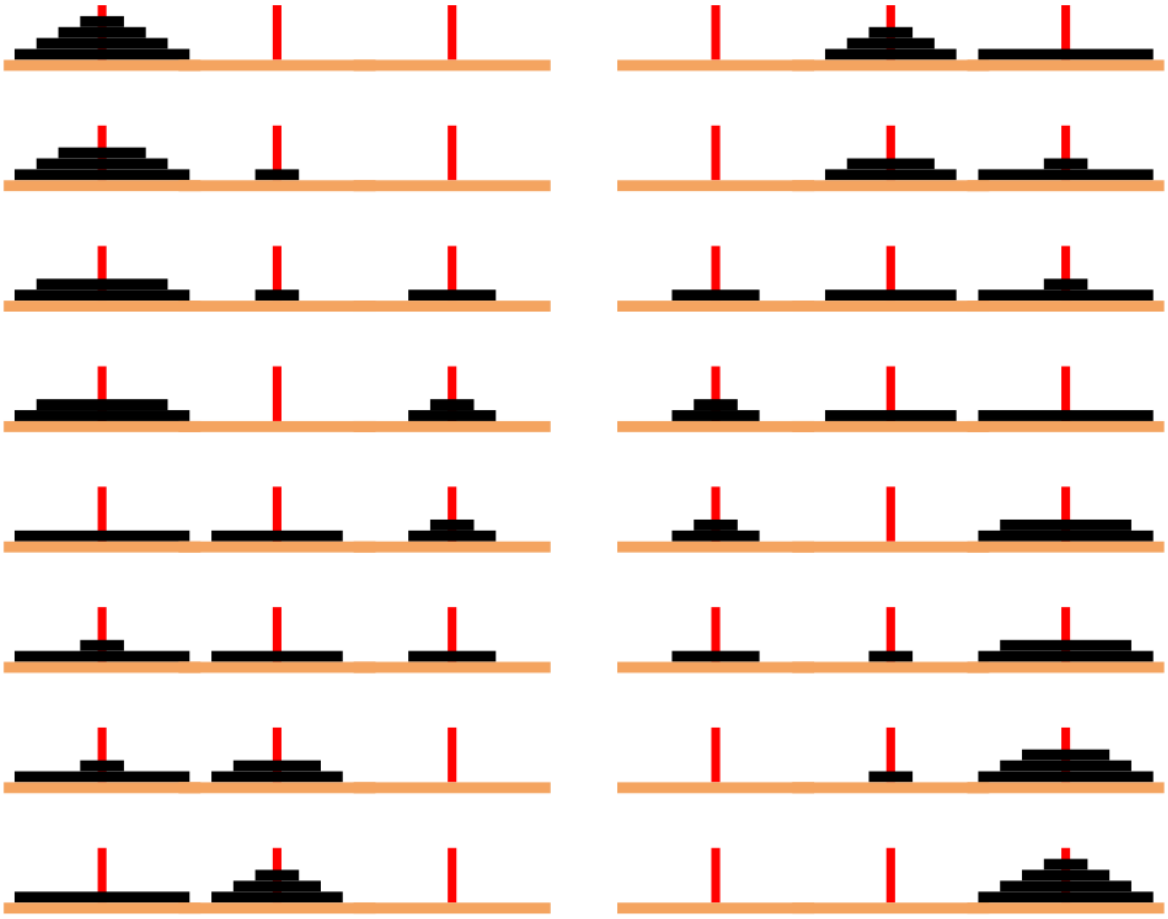

*Image credit: Wolfram MathWorld