Problem set `hw:a2`

Class (One Day)

1. (Sum Square Difference)[https://projecteuler.net/problem=6] The sum of the squares of the first ten natural numbers is,

   `1^2 + 2^2 + ... + 10^2 = 385`

   The square of the sum of the first ten natural numbers is,

   `(1 + 2 + ... + 10)^2 = 55^2 = 3025`

   Hence the difference between the sum of the squares of the first ten natural numbers and the square of the sum is $3025 - 385 = 2640$.

   Find the difference between the sum of the squares of the first one hundred natural numbers and the square of the sum.

2. (Multiples of 3 and 5)[https://projecteuler.net/problem=1] If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23. Find the sum of all the multiples of 3 or 5 below 1000.

3. (Smallest Multiple)[https://projecteuler.net/problem=5] 2520 is the smallest number that can be divided by each of the numbers from 1 to 10 without any remainder. What is the smallest positive number that is evenly divisible by all of the numbers from 1 to 20?

Homework

4. Write the "grapher" function whose inputs are (i) a function f that takes in a number and puts out a number, (ii) a lower bound, and (iii) an upper bound. The grapher function should return a list of coordinates `(x,y)` where `y=f(x)` and `x` goes from the lower bound to the upper bound increasing by 1 each time.

5. Write a function `onLine` that takes in a slope `m`, a y-intercept `b`, and a point `(x0,y0)` and gives back True if the point is on the line `y=m*x+b` and false otherwise.

6. Write a function `moreThan100` that takes in a list of numbers and puts out a list of numbers. The output is the same as the input with all of the numbers less than or equal to 100 removed.

7. *Break It Up.* Given a list of numbers, create a list containing every sequence of four numbers in a row. Example:

   breakItUp [5,10,20,3,8,9] = [[5,10,20,3], [10,20,3,8], [20,3,8,9]]

8. *Target Practice.* (*Challenge*) Given a point `(x0,y0)` and a list of ordered pairs, return the point in the list that is closest to the given point. (Hint: skip the square root when you compare distances, it will save you some Haskell difficulties.)

OMIT

9. Write a function `addp` that adds two ordered pairs.
10. (*Bonus*) The function `nextInDirection` takes in start point `(x0,y0)`, a direction `(dx,dy)`, and a list of ordered pairs. The function returns the point in the list that is closest to the start point and can be reached by beginning at the start and going in the given direction.