CS 5035 (Fall 2016)

### Project 2. Types and typeclasses (first attempt by Sept 7)

Based on chapter [3 of LYH](http://learnyouahaskell.com/types-and-typeclasses). [Videos](https://sites.google.com/a/lclark.edu/drake/courses/pls/lesson-2-haskell-types-and-typeclasses).

Goldbach’s conjecture is that every even number > 2 can be expressed as the sum of two primes. (See, for example, [Wikipedia](https://en.wikipedia.org/wiki/Goldbach%27s_conjecture).) This has never been proven, but no counterexamples are known.

Goldbach’s other conjecture is that every odd number > 2 can be expressed as the sum of a prime and twice the square of some other number. In other words, for every odd number g > 2 there is a prime p and some other number k such that g = p + 2 \* k \* k. (See “[Toying with a lesser known Goldbach conjecture](http://brainwagon.org/2006/09/05/toying-with-a-lesser-known-goldbach-conjecture/).” The article notes that besides 5777 and 5993, 3, 17, and a number of others also cannot be expressed as p + 2 \* k \* k as long as we require that k > 0.)

The following code shows that Goldbach was wrong about this other conjecture. Be prepared to explain this code and to answer the embedded questions.

oddsFrom3 = [3, 5 .. ]

{-

**Question**: What are the differences among these lists?

a) listA = [3, 5 ..]  
b) listB = [x | x <- [3, 5 ..]]  
c) listC = [x | x <- [3, 5 .. 5999]]  
d) listD = [x | x <- [3, 5 .. 6000]]  
e) listE = [x | x <- [3, 5 ..], (x < 6000)]  
f) listF = [x | x <- takeWhile (< 6000) [3, 5 ..] ]

In other words, for any pair of the preceding lists, list\_a and list\_b

a) is it true that list\_a == list\_b? (Why or why not?)

b) will Haskell return True for

> list\_a == list\_b

-- Why or why not?

-}

<=2 [2,3], 4 % 2 == 0

primeDivisors n = [d | d <- takeWhile ((<= n) . (^2)) primes, n `mod` d == 0]

primes = 2 : [p | p <- oddsFrom3, null (primeDivisors p)]

-- **Approach 1**. Look for “k” values.  
isPrime g = g == head (dropWhile (< g) primes)

-- **Question**: How does isPrime work?

-- Generates for any number g the p and k such that g = p + 2 \* k \* k  
goldbachPairs g = [(p, k) | k <- takeWhile ((< g) . (\*2) . (^2)) [1 .. ],   
 p <- [g - 2 \* k \* k], isPrime p]

-- **Question**: What does “p <- [g - 2 \* k \* k]” do in the preceding?

-- **Question**: Explain in your own words what goldbachPairs g contains.

goldbachFails = [g | g <- takeWhile (< 6000) oddsFrom3, not (isPrime g),   
 null (goldbachPairs g) ]

-- **Question**: What is the list goldbachFails?

-- **Approach 2**. Look for “p” values.  
iSqrt n = floor (sqrt (fromIntegral n))

isSquare n = (iSqrt n) ^ 2 == n

-- **Question**: Explain how iSqrt works?

goldbachPairs' g = [(p, iSqrt kSqr) | p <- takeWhile (< g) (tail primes),   
 kSqr <- [(g - p) `div` 2],   
 isSquare kSqr]

-- **Question**: What does “kSqr <- [(g - p) `div` 2]” do in the preceding?

-- **Question**: Explain in your own words what goldbachPairs’ g contains.

-- **Question**: Why does the search look through (tail primes) rather than primes?

goldbachFails' = [g | g <- takeWhile (< 6000) oddsFrom3, not (isPrime g),   
 null (goldbachPairs' g) ]

Determine and explain the types of the functions in the code above.

See [Haskell Types and Typeclasses](https://docs.google.com/document/d/1ognQt40bYDfvjjJZLuZKARTo0HttIv1mx_HQo5_slBQ/) for a summary overview. Read [Converting numbers](https://wiki.haskell.org/Converting_numbers) for information on Haskell number types.

Note that with no type declarations

> :t isSquare

isSquare :: Integral a => a -> Bool

-- **Question**: Explain the preceding type declaration.

> :t primes

primes :: [Integer]

If you declare

isSquare :: Int -> Bool

> :t primes

primes :: [Int]

Why is that?

Suppose you define

goldbachDiffs = [g | g <- takeWhile (< 6000) oddsFrom3, not (isPrime g), goldbachPairs g /= reverse (goldbachPairs' g)]

What do you suppose you would get if you ran

> goldbachDiffs

-- **Question**: Explain why you get that result.