CS 5035 (Fall 2016)

### Project 1. Introduction and starting out (first attempt by Aug 29)

Based on chapters [1](http://learnyouahaskell.com/introduction) and [2](http://learnyouahaskell.com/starting-out) of [LYH](http://learnyouahaskell.com/chapters). V[ideos](https://sites.google.com/a/lclark.edu/drake/courses/pls/lesson-1-haskell-introduction-and-starting-out).

Use list comprehension to generate [the 2 non-prime odd numbers less than 6,000 not of the form p + 2 \* k^2, where p is prime and k>0](http://oeis.org/A060003) . (The answer: [5777, 5993].) This is known as [Goldbach’s other conjecture](https://projecteuler.net/problem=46).

As a head start, here is some code that generates the prime numbers.

The following code uses the function takeWhile, not covered in chapters 1 or 2. takeWhile takes elements from a list while some predicate holds. For example:

> takeWhile (> 4) [8, 7, 6, 5, 4, 3, 2, 3, 4, 5, 6, 7, 8]

[8,7,6,5]

Here is some code to generate the prime numbers.

-- A lazy list of odd numbers beginning at 3

oddsFrom3 = [3, 5 .. ]

-- The prime divisors of n. The predicate to takeWhile is the   
-- composition (denoted “.”) of two functions. It succeeds when the   
-- square of the number is less than or equal to n.  
-- Note that primeDivisors and primes are defined in terms of each other.

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

-- The prime numbers: 2 followed by the odd primes.

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

> take 25 primes

[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97]

**Hints**

1. For each odd non-prime g < 6000 determine if there is a p and k such that  
   g = p + 2 \* k^2. If there is no p and k, g is one of the numbers you are looking for. If you try all combinations of p and k, your program will run for a very long time. Instead, look for a prime p such that (g - p)/2 is a square.
2. Write a predicate isASquare that determines whether its argument is a square.
3. You will also find it useful to write a predicate isPrime. What’s wrong with this?

isPrime n = n `elem` primes

Ask yourself how long the list primes is.

1. An alternative to looking for a p, as in (1), is to look for a k. Find a k such that g - 2\*k^2 is prime. Where should you look for such a k? Look in [1 .. ] as long as 2\*k^2 < g.