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]

In the code below, takeWhile (<= p `div` 2) ints2 returns numbers between 2 and p `div` 2, where `div` means integer division.

ints2 = [2 .. ]

smallDivs n = [d | d <- takeWhile (<= n `div` 2) ints2, n `mod` d == 0]

primes = [p | p <- ints2, smallDivs p == []]

**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 function isASquare that returns True or False depending on whether its argument is a square.
3. You will also find it useful to write a function isPrime that returns True or False depending on whether its argument is prime. What’s wrong with this?

isPrime n = n `elem` primes

n `elem` primes asks whether n is an element of primes. Ask yourself how long the list primes is.