CS 430 Programming Languages
Spring 2017
Module 10 PA

JAMES MADISON UNIVERSITY

Introduction

In this PA we will simply be practicing writing some Haskell functions. I have prepared a template file called mod10PATemplate.hs. Please download this file and rename it mod10PA.hs. Then open the file and write your code in it, filling out the stubbed functions.

The Tests

If you look towards the bottom of mod10PATemplate.hs you will see a bunch of test functions. You should not change these functions in any way. If you load this file into ghci as is and type test, you will see that all the tests fail. The idea is that as you write your code, you can keep running test (or any of the other test functions) and determine whether your code is (probably) correct. I will also run test to see if you code is correct.

Do NOT use any Haskell prime number packages (such as Data.Numbers.Primes).

The Functions

The functions you need to write are stubbed out in the file. Here are descriptions of them (you can also look at the test cases to see what they are supposed to do, of course).

factors :: Integral a => a -> [a]. This function must take a non-negative whole number and return a list of all its factors, including the number itself. Its result must be the empty list if its argument is zero or negative.

isPrime :: Integral a => a -> Bool. This function must take a non-negative whole number and return True if it is prime and False otherwise. Its result must be False if its argument is zero or negative.

primeFactors :: Integral a => a -> [a]. This function must take a non-negative whole number and return a list of all its prime factors, possibly including the number itself. Note that 1 is not a prime number. Its result must be the empty list if its argument is zero or negative.

primesUpTo :: Integral a => a -> [a]. This functions must take a non-negative whole number and return a list of prime numbers up to and possibly including the argument. Its result must be the empty list if its argument is zero or negative.

isPerfect :: Integral a => a -> Bool. A perfect number is a number that is half the sum of its factors. For example, the factors of 6 are 1, 2, 3, and 6, whose sum is twice six, or 12. isPerfect must take a non-negative whole number and return True if its argument is a perfect number and False otherwise. Its result must be False if its argument is zero or negative.

perfectUpTo :: Integral a => a -> [a]. This function must take a non-negative whole number and return a list of all perfect numbers up to and possibly including its argument. Its result must be the empty list if its argument is zero or negative.

`pythagoreans :: Integral a => a → [(a,a,a)]`. This function must take a non-negative whole number n and return a list of all triples (x,y,z) such that $x^2 + y^2 = z^2$, where $x, y, z \leq n$, and $x \leq y < z$. In other words, it returns whole numbers constituting the sides or right triangles, such as (3,4,5). Its result must be the empty list if its argument is zero or negative.

`nextPrime :: Integral a => a -> a`. This function must take a whole number and return the first prime greater than its argument. For example, the first prime greater than 6 is 7, and the first prime greater than 7 is 11. Note that since the first prime number is 2, this function must always return 2 for any number less than 2.

`generatePrimes :: Integral a => a -> [a]`. This function must take a non-negative whole number `n` and return a list of the first `n` primes. For example, the first three primes are 2, 3, and 5. This function must return the empty list for all `n` less than 1.

These functions are arranged in order with the idea that you can use functions that you have previously implemented to implement functions later in the template (of course, you don't have to do this).

Deliverable Requirement

Your program must be named `mod10PA.hs`. This file must have test code at the bottom unchanged from `mod10PATemplate.hs`.

Submit your Haskell file in Canvas by the listed due date.