課題 2

n 番目の素数を求める関数 nthPrime を定義しなさい。

解答及び解説

基本的な考えに基づく解答は、次のようなものになる。

```
nthPrime :: Integer -> Integer
nthPrime 1 = 2
nthPrime n = nextPrime ((nthPrime (n-1)) + 1)

nextPrime :: Integer -> Integer
nextPrime n | prime n   = n
            | otherwise = nextPrime (n+1)
```

但し、これでは冗長な部分が多く、効率があまり良くない。少なくとも、「2 以外の素数は、奇数である」ということを前提に素数を探すようにするのが良い。

```
nthPrime :: Integer -> Integer
nthPrime 1 = 2
nthPrime 2 = 3
nthPrime n = nextPrime ((nthPrime (n-1)) + 2)

nextPrime :: Integer -> Integer
nextPrime n | prime n   = n
            | otherwise = nextPrime (n+2)
```

また、与えられた数 n が素数かどうかを判定する述語 prime が必要である。これをどのように定義するかも考える必要がある。

まずは、「素数の定義」をそのまま関数の定義として利用する方法。つまり「素数とは、1 と自分自身以外に約数を持たない自然数」

そこで、2 から始めて、n を割ってみる。

a) 割り切れたときは、n は素数では無い

b) 割り切れないときは、次の数を試す

c) 割ってみる数が、n になったら、n は素数である

```
prime :: Integer -> Bool
prime n = prime' n 2
```

```
prime' :: Integer -> Integer -> Bool
prime' n m | n == m       = True
           | mod n m == 0 = False
           | otherwise    = prime' n (m+1)
```

もちろん、試すのも奇数に限ることができるので、

```
prime :: Integer -> Bool
prime 2 = True
prime n | mod n 2 == 0 = False
        | otherwise    = prime' n 3

prime' :: Integer -> Integer -> Bool
prime' n m | n == m       = True
           | mod n m == 0 = False
           | otherwise    = prime' n (m+2)
```

実際には、2 以上、$\sqrt{n}$ 以下の奇数で割ってみれば良い

```
prime' :: Integer -> Integer -> Bool
prime' n m | n < m*m       = True
           | mod n m == 0 = False
           | otherwise    = prime' n (m+2)
```

但し、$\sqrt{n}$ を計算するのは大変なので、割る数を 2 乗して n と比べる。さらには、る数は素数に限って良いので、次のように素数のリストを作り、それを用いて素数を判定することも可能である。

```
primes :: [Integer]
primes = 2:3:map nextPrime (tail primes)

nextPrime :: Integer -> Integer
nextPrime n | prime m   = m
            | otherwise = nextPrime m
    where m = n + 2

prime n :: Integer -> Bool
prime n = prime' n primes

prime' :: Integer -> [Integer] -> Bool
```

```
prime' n (m:ms) | n < m*m      = True
                | mod n m == 0 = False
                | otherwise    = prime' n ms
```

素数のリスト primes を用いれば、nthPrime は次のようになる。

```
nthPrime :: Int -> Integer
nthPrime n = primes!!(n-1)
```

(n-1) とするのは、リストが 0 番から始まるからである。