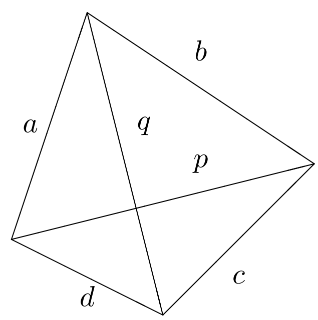
Haskell Answers 3: Floating-point Numbers

Antoni Diller

4 August 2011

(1) The area of a general quadrilateral



is given by the formula

$$area = \frac{1}{4}\sqrt{4p^2q^2 - (b^2 + d^2 - a^2 - c^2)^2},$$

where a, b, c and d are the lengths of the sides and p and q are the lengths of the diagonals. Defines a Haskell function *area* $a\ b\ c\ d\ p\ q$ which calculates the area of a general quadrilateral. Your function should return an error message if the arguments given to it are not those of a genuine quadrilateral. (Hint: test whether a, b and p really are the lengths of three sides of a triangle, etc.)

```
sort :: Ord a => [a] -> [a]
sort [] = []
sort [x] = [x]
sort (x:xs) = sort [y | y <- xs, y < x] ++ [x] ++ sort [y | y <- xs, y >= x]

sort3 :: Ord a => (a,a,a) -> (a,a,a)
sort3 (x, y, z) = (p, q, r)
                  where p = head ws
                        q = head (tail ws)
                        r = last ws
                        ws = sort [x, y, z]

testTriangle :: (Num a, Ord a) => (a,a,a) -> Bool
-- x <= y <= z
testTriangle (x, y, z)
  | x + y < z = error "Non-triangle encountered"
  | otherwise = True

area :: (Ord a, Floating a) => a -> a -> a -> a -> a -> a -> a
area a b c d p q
  | testTriangle (sort3 (a, b, p)) &&
    testTriangle (sort3 (b, c, q)) &&
    testTriangle (sort3 (c, d, p)) &&
    testTriangle (sort3 (a, d, p))
     = sqrt (4*p*p*q*q - square (b*b + d*d - a*a - c*c)) / 4
  | otherwise                      = error "Impossible error"

square :: Num a => a -> a
square x = x*x
```

(2) The Fibonacci numbers are usually defined as follows:

$$f(1) = 1,$$
$$f(2) = 1,$$
$$f(i) = f(i-1) + f(i-2), \text{if } i > 2.$$

They can, however, be defined as follows:

$$f(i) = \frac{1}{\sqrt{5}} \left\{ \left(\frac{1+\sqrt{5}}{2} \right)^i - \left(\frac{1-\sqrt{5}}{2} \right)^i \right\}.$$

Define a Haskell function *fd* (Fibonacci direct) that uses this formula. Use a local definition to avoid recalculation of $\sqrt{5}$ and ensure that the answer is an integer.

```
fd :: (Floating a, Num b) => b -> a
fd i = ((p - q) / sqrt 5)
       where p = power ((1 + sqrt 5) / 2) i
             q = power ((1 - sqrt 5) / 2) i

fdp :: (Num a, Floating b) => a -> b
fdp i = power ((1 + sqrt 5) / 2) i

fdq :: (Num a, Floating b) => a -> b
fdq i = power ((1 - sqrt 5) / 2) i

power :: (Num a, Num b) => a -> b -> a
power x 1 = x
power x y = x * power x (y-1)
```

(3) To decide which weekday a certain date is, you can use the formula $v = A \mod 7$ where $A = \lfloor 2.6 \times m - 0.2 \rfloor + d + y + \lfloor \frac{y}{4} \rfloor + \lfloor \frac{c}{4} \rfloor - 2 \times c$. Here, $\lfloor x \rfloor$ is the integral part of x. For example, $\lfloor 2.7 \rfloor = 2$. Furthermore, v is the weekday (with Sunday as day 0), d is the day of the month, m is the month (with March being 1 and February 12), y is the last two digits of the year and c is the century. What is the weekday of the following dates: 10 July 1776, 16 April 1834 and 9 June 1901?

```
weekday :: RealFrac a => Int -> a -> Int -> Int
weekday dd mm yy
  = (floor (2.6*mm - 0.2) + dd + y +
     floor (fromInt y/4) +
     floor (fromInt c/4) - 2* fromInt c) `mod` 7
     where y = yy `rem` 100
           c = yy `div` 100
```

(4) The harmonic series is the following infinite series:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \ldots + \frac{1}{i} + \ldots .$$

Write a function *sumHarmonic* such that *sumHarmonic* i is the sum of the first i terms of this series. For example, *sumHarmonic* $4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} = 2.08333\ldots$.

```
sumHarmonic :: Fractional a => a -> a
sumHarmonic 1 = 1
sumHarmonic i = 1/i + sumHarmonic (i-1)
```

(5) The logarithmic series is the following alternating series:

$$1 - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + (-1)^{n-1}\frac{x^n}{n} + \ldots \ .$$

Write a function *sumLog* such that *sumLog* n x is the sum of the first n terms of this series.

```
sumLog :: Double -> Double -> Double
sumLog 1 x = x
sumLog n x = ( power (-1) (n-1) ) * ( (x ** n) / n ) + sumLog (n-1)  x
```