```haskell
-- Caeser cipher example from chapter 5 of Programming in Haskell,
-- Graham Hutton, Cambridge University Press, 2016.

import Data.Char

-- Encoding and decoding

let2int :: Char -> Int
let2int c = ord c - ord 'a'

int2let :: Int -> Char
int2let n = chr (ord 'a' + n)

shift :: Int -> Char -> Char
shift n c | isLower c = int2let ((let2int c + n) `mod` 26)
          | otherwise = c

encode :: Int -> String -> String
encode n xs = [shift n x | x <- xs]

-- Frequency analysis

table :: [Float]
table = [8.1, 1.5, 2.8, 4.2, 12.7, 2.2, 2.0, 6.1, 7.0,
         0.2, 0.8, 4.0, 2.4, 6.7, 7.5, 1.9, 0.1, 6.0,
         6.3, 9.0, 2.8, 1.0, 2.4, 0.2, 2.0, 0.1]

lowers :: String -> Int
lowers xs = length [x | x <- xs, x >= 'a' && x <= 'z']

count :: Char -> String -> Int
count x xs = length [x' | x' <- xs, x == x']

percent :: Int -> Int -> Float
percent n m = (fromIntegral n / fromIntegral m) * 100

freqs :: String -> [Float]
freqs xs = [percent (count x xs) n | x <- ['a'..'z']]
           where n = lowers xs

chisqr :: [Float] -> [Float] -> Float
chisqr os es = sum [((o-e)^2)/e | (o,e) <- zip os es]

rotate :: Int -> [a] -> [a]
rotate n xs = drop n xs ++ take n xs

positions :: Eq a => a -> [a] -> [Int]
positions x xs = [i | (x',i) <- zip xs [0..n], x == x']
                 where n = length xs - 1

crack :: String -> String
crack xs = encode (-factor) xs
           where
              factor = head (positions (minimum chitab) chitab)
              chitab = [chisqr (rotate n table') table | n <- [0..25]]
              table' = freqs xs
```