import Char

-- 1

f :: [Int] -> Bool
f xs = and [ x 'mod' 2 == 0 | x <- xs, x > 0 ]

g :: [Int] -> Bool
  g [] = True
  g (x:xs) | x > 0 = x 'mod' 2 == 0 && g xs
  | otherwise = g xs

h :: [Int] -> Bool
  h = foldr (&&) True . map ((== 0) . ('mod' 2)) . filter (>0)

test1 = ok f && ok g && ok h
  where
    ok f = f [2,10,8] && not (f [2,9,8]) && f [2,-9,8]

-- 2

p :: Int -> Int -> Int
  p i j | i > j = i-j
  | i == j = 0
  | j > i = j-i

q :: [Int] -> Int
  q (i:is) = sum [ p i j | (i,j) <- zip is (i:is) ]

r :: [Int] -> Int
  r [i] = 0
  r (i:j:js) = p i j + r (j:js)

test2 = ok q && ok r
  where
    ok q = q [1,2,4,7,3,8] == 15 &&
              q [8,3,7,4,2,1] == 15 &&
              q [1,2,3,4,5,6] == 5 &&
              q [6,5,4,3,2,1] == 5 &&
              q [3,3,3,3,3] == 0

-- 3

t :: Int -> Int -> [a] -> [a]
  t i j = take (j-i) . drop i

t' :: Int -> Int -> [a] -> [a]
  t' i j xs = [ x | (k,x) <- zip [0..] xs, i <= k, k < j ]

u :: Int -> Int -> [a] -> [a]
  u 0 0 xs = []
  u 0 (j*i) (x:xs) = x : u 0 j xs
\[(i+1) \cdot (j+1) \cdot (x:xs) = u \cdot i \cdot j \cdot xs\]

test3 = ok t \&\& ok t' \&\& ok u

where

\[
\begin{align*}
ok t &= t 0 6 \text{"abcdef"} == \text{"abcdef"} \&\& \\
& t 1 5 \text{"abcdef"} == \text{"bcde"} \&\& \\
& t 2 4 \text{"abcdef"} == \text{"cd"} \&\& \\
& t 3 3 \text{"abcdef"} == \text{""} \\
\end{align*}
\]

test = test1 \&\& test2 \&\& test3