import Char

-- 1

-- 1a

f :: [Int] -> Int
f xs = product [ x+1 | x <- xs, 3 <= x, x <= 7 ]

-- 1b

g :: [Int] -> Int
g [] = 1
g (x:xs) | 3 <= x && x <= 7 = (x+1) * g xs
      | otherwise = g xs

-- 1c

h :: [Int] -> Int
h = foldr (*) 1 . map (+1) . filter (3 <=) . filter (<= 7)

test1 = ok f && ok g && ok h
  where
    ok f = f [0,3,8,-42,7,1,4] == 160

-- 2

-- 2a

p :: Char -> Char -> Bool
p x y = (isAlpha x && not (isAlpha y)) || (not (isAlpha x) && isAlpha y)

-- 2b

q :: String -> Int
q (x:xs) = length [ () | (x,y) <- zip (x:xs) xs, p x y ] + 1

-- 2c

r :: String -> Int
r [x] = 1
r (x:y:zs) | p x y = 1 + r (y:zs)
          | otherwise = r (y:zs)

test2 = ok q && ok r
  where
ok q =
  q "Hello, world!" == 4 &&
  q "Hello, world" == 3 &&
  q "Phil’s #*! class" == 5

-- 3
-- 3a

t :: Int -> [a] -> [a]
t n xs = [ xs !! (i 'mod' length xs) | i <- [0..n-1] ]

-- or

t' :: Int -> [a] -> [a]
t' n xs = take n (cycle xs)
  where
    cycle xs = xs ++ cycle xs

-- 3b

u :: Int -> [a] -> [a]
u n xs = v n xs
  where
    v 0 ys = []
    v i [] = v i xs
    v (i+1) (y:ys) = y : v i ys

test3 = ok t && ok t' && ok u
  where
    ok t =
      t 0 "abcd" == "" &&
      t 2 "abcd" == "ab" &&
      t 4 "abcd" == "abcd" &&
      t 6 "abcd" == "abcdab" &&
      t 8 "abcd" == "abcdabcd" &&
      t 17 "abcd" == "abcdabcdabcdabcdab"