Q 1

a). i. sum $ take k powers

ii. length $ takeWhile $(< t\theta)$ powers
(with superscript k over $t\theta$)

iii. sum $ takeWhile $(< k)$ powers

iv. dropWhile $(< j)$ $((takeWhile (\leq k) (powers)))$

b). i. pairs $n = [(x,y) \mid x \in [1..n-1], y \in [x+1..n]]$

c).
```
smallWords :: String -> Int
smallWords s = length [x | x <- words s, length x < 4]
```

d).
```
printLines :: FilePath -> IO ()
printLines = do
    contents <- readFile "in.txt"
    putStr ( unlines $ map (\n -> length x ++
        " " )xs )xs
  where
    xs = lines contents
```
could also
enter file name

2.

a). rotate x = tail x ++ [head x]

b). mystery x ys = takeWhile (<= x) ys

c).

i. $[x^2 \mid x \leftarrow xs, x \text{ `mod` } 2 == 1]$

ii. $[x*5 \mid x \leftarrow xs, 2^x > 1000]$

(d). replace :: Int -> Int -> [Int] -> [Int]
replace x y zs = [if z == x then y else z $\mid$ z $\leftarrow$ zs]

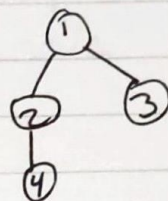(e). compose :: [a -> a] -> a -> a

Fn

compose [] x = x
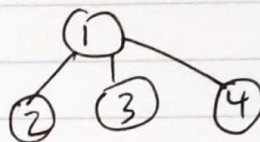compose (F : fn) x = F (compose fn x)

Question 3.

(a). Node 1 [] ①

Node 1 [Node 2 [Node 4 [ ]],
         Node 3 [ ]]

Node 1 [Node 2 [ ],
         Node 3 [ ], Node 4 [ ]]

(b). leaf :: a → Tree a
     leaf x = Node x [ ]

(c) complex :: Tree a → Bool
    complex (Node _ x) = x /= [ ]

(d) productTree :: Tree Int → Int
    productTree (Node x ts) = x * product
              (map productTree ts)

(e). foldr.
     last leaf :: Tree a → a
     last leaf (Node x [ ]) = x 0
     last leaf (Node x ts) =

```haskell
2.    prune :: (a -> Bool) -> Tree a -> Tree a

Prune f (Node a x) = if (F a) then (Node a [])
        else (Node a (prune F x))
```