import Data.Char

import Data.List

import System.Environment

import System.IO

minus = map toLower

removPonto [] = []

removPonto (x:xs) = if not(isDigit x) && not(isAlpha x) && not(isSpace x) then removPonto xs

else x:removPonto xs

main = do

aa <- getLine

arq <- readFile aa

let res = remove3(eliminarRep(agrupar(ordenar(numeraPalavras(numLinhas (lines (minus ( removPonto arq))))))))

print res

-- a) Separar o documento em linhas (lines biblioteca haskell)

--b) Numerar as linhas do documento

numLinhas' \_ [] = []

numLinhas' n (x:xs) = (n,x):numLinhas'(n+1) xs

numLinhas xs = numLinhas' 1 xs

--c) Associar a cada ocorrência de uma palavra do documento, o número da linha em que essa palavra ocorre

numeraPalavras [] = []

numeraPalavras ((num, linha):xs) = zip (repeat num) (words linha) ++ numeraPalavras xs

--d)Ordenar alfabeticamente as ocorrências das palavras no texto

inverte [] = []

inverte ((x,y):xs) = (y,x):inverte xs

ordenar x = inverte(sort(inverte x))

{--e)Juntar as várias ocorrências de cada palavra, produzindo, para cada palavra, a lista dos

números das linhas em que a palavra ocorre--}

iguais (x,y) [] = [x]

iguais (x,y) ((a,b):xs) = if y == b then a: iguais (x,y) xs else iguais (x,y) xs

agrupar [] = []

agrupar ((x,y):xs) = (iguais (x,y) xs, y): agrupar (filter (\(a,b) -> b /= y) xs)

{--f) Eliminar, da lista de números de linhas em que cada palavra ocorre,

as repetições de um mesmo número de linha:--}

possui n [] = False

possui n (x : xs) = if n == x then True else possui n xs

rep [] = []

rep (x : xs) = if possui x xs then rep xs else x : rep xs

eliminarRep [] = []

eliminarRep (x : xs) = ((rep (fst x)), snd x) : eliminarRep xs

remove3 [] = []

remove3 ((x,y):xs) = if length y <3 then remove3 xs else (x,y): remove3 xs