```haskell
{- *
   * Find all of the words in the dictionary that have two or more
   * consecutive pairs of letters.
-}

module Main
where

import System.IO    -- openFile ReadMode
import Data.Char    -- toLower

runs :: Eq a => [a] -> [(Int,a)]
runs [] = []
runs (x:xs) = runs' (1,x) xs where
  runs' a [] = [a]
  runs' (n,x) (y:ys)
    | x==y          = runs' (n+1,x) ys
    | otherwise     = (n,x) : runs' (1,y) ys

successivePairsCounts :: [Int] -> [Int]
successivePairsCounts (2:xs) = let
  (ys,zs) = span (==2) xs
  in 1+length ys: successivePairsCounts zs
successivePairsCounts (_:xs) =
  successivePairsCounts . dropWhile (/= 2) $ xs
successivePairsCounts [] = []

maxPairCounts :: Eq a => [a] -> Int
maxPairCounts xs = maximum . (0:) . successivePairsCounts . map fst . runs $ xs

bookkeeperIsh :: String -> Bool
bookkeeperIsh = (> 3) . maxPairCounts


getFileContents :: String -> IO String
getFileContents dictFile = do
  dictHandle   <- openFile dictFile ReadMode
  fileContents <- hGetContents dictHandle
  return fileContents

dictFileName :: String
dictFileName = "/Users/casper/Downloads/finnish-words-kaikkisanat.txt"

main :: IO ()
main = do
  fileContents <- getFileContents dictFileName
  putStr . unlines . filter bookkeeperIsh . lines $ fileContents
```