```haskell
import System.Environment
import System.Random
import Data.List

average :: [Integer] -> Double
average [] = error "empty list"
average xs = fromIntegral (sum xs) / fromIntegral (length xs)

removeLessThan :: Ord a => [a] -> a -> [a]
removeLessThan xs value = filter ((<) value) xs

randomList :: Int -> StdGen -> [Int]
randomList n = take n . unfoldr (Just . random )

{- Usage:
        ghc AveList.hs
        ./AveList 500 3.0e8
-}
main = do
    (a:b:unused) <- getArgs
    seed <- newStdGen
    let n = (read a) :: Int
    let threshold = (read b) :: Double
    let numList = randomList n seed
    let numList'= removeLessThan numList (round threshold)
    print $ average $ map fromIntegral numList'
    -- or
    print (average (map fromIntegral numList'))
```