```haskell
module Assist_Lib
( store
, pack
, tail'
, head'
, init'
, last'
, isSquare
, isVowel
, isConsonant
, points
, someNumbers
, someLetters
, trimListOfKey
) where

import           System.Random

tail' :: [a] -> [a]
tail' [] = []
tail' (_:theList) = theList

head' :: [a] -> a
head' (theHead:_) = theHead

init' :: [a] -> [a]
init' [] = []
init' [x] = []
init' (x:y:xs) = x : init' (y:xs)

last' :: [a] -> a
last' [x] = x
last' (_:x:xs) = last' (x:xs)

deleteMap = randoms (mkStdGen 1) :: [Bool]
someLetters = randomRs ('a', 'z' ) (mkStdGen 1) :: [Char]
someNumbers = randomRs (-100, 100) (mkStdGen 1) :: [Int]

store :: [a] -> [(a, Bool)]
store [] = []
store theData = zip theData deleteMap

pack :: [(a, Bool)] -> [(a, Bool)]
pack dataBlocks = [ validData
      | validData <- dataBlocks,
        not $ snd validData]

points = [(4, 3), (1, 6), (2, 4)]

isSquare :: (Integral a) => a -> Bool
isSquare num = num == (^2) (floor $ sqrt $ fromIntegral num)

isVowel :: Char -> Bool
isVowel c = elem c ['a', 'e', 'i', 'o', 'u']

isConsonant :: Char -> Bool
isConsonant c = not $ isVowel c

trimListOfKey :: (Eq a) => a -> [(a, b)] -> [(a, b)]
trimListOfKey key = filter (\x -> fst x /= key)
```