Zadanie N: Haskell - Randomizowany Quick Sort

Typ *Randomised* definiujemy następująco:

```
newtype Randomised a = Rand (StdGen->(a, StdGen))
```

Typ *StdGen* pochodzi z System.Random. Uwaga: Ubuntu postanowiło okroić Haskella o System.Random. Na Satori System.Random jest, natomiast aby z niego korzystać na maszynach uczelnianych najprościej wgrać dołączony plik do System/Random.hs (uwaga na podkatalog) w katalogu w którym uruchamiamy program.

Uczyń typ *Randomised* instancją klasy typów *Monad*. Napisz funkcję *solution* typu:

```
(Ord a)=> [a]-> Randomised [a]
```

implementującą randomizowany algorytm QuickSort, (randomizowany jest wybór elementu dzielącego).

Niech *runR* będzie destruktorem typu Randomised, czyli:

```
runR (Rand rf) = rf
```

Wtedy dla dowolnej listy (porównywalnych elementów) lst oraz generatora sg, wartością fst (runR (solution lst) sg) będzie będzie posortowana lista lst.

W pliku z rozwiązaniem musi się znaleźć następująca definicja typu *Randomised*:

```
module Solution where
import System.Random
newtype Randomised a = Rand (StdGen->(a, StdGen))
instance ...
solution :: (Ord a) => [a] -> Randomised [a]
solution = ...
```

Uwaga! Po akceptacji przez system programy mogą zostać odrzucone przez prowadzącego, jeśli implementacja nie będzie monadyczna lub jeśli algortym nie będzie randomizowany. Przykład

```
module Main where

import System.Random
import Solution

main= do
  i1<- getContents
  let l1= (read i1)::[Int]
  s1<- getStdGen
  let (Rand qs)= solution l1
  print$fst$qs s1
```