1 Problem

In the card game poker, a hand consists of five cards and are ranked, from lowest to highest, in the following way:

- **High Card:** Highest value card.
- **One Pair:** Two cards of the same value.
- **Two Pairs:** Two different pairs.
- **Three of a Kind:** Three cards of the same value.
- **Straight:** All cards are consecutive values.
- **Flush:** All cards of the same suit.
- **Full House:** Three of a kind and a pair.
- **Four of a Kind:** Four cards of the same value.
- **Straight Flush:** All cards are consecutive values of same suit.
- **Royal Flush:** Ten, Jack, Queen, King, Ace, in same suit.

The cards are valued in the order: 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen, King, Ace.

If two players have the same ranked hands then the rank made up of the highest value wins; for example, a pair of eights beats a pair of fives (see example 1 below). But if two ranks tie, for example, both players have a pair of queens, then highest cards in each hand are compared (see example 4 below); if the highest cards tie then the next highest cards are compared, and so on.

Consider the following five hands dealt to two players:

Hand	Player 1	Player 2	Winner
1	5H 5C 6S 7S KD	2C 3S 8S 8D TD	Player 2
	Pair of Fives	Pair of Eights	
2	5D 8C 9S JS AC	2C 5C 7D 8S QH	Player 1
	Highest card Ace	Highest card Queen	
3	2D 9C AS AH AC	3D 6D 7D TD QD	Player 2
	Three Aces	Flush with Diamonds	
	4D 6S 9H QH QC	3D 6D 7H QD QS	
4	Pair of Queens	Pair of Queens	Player 1
	Highest card Nine	Highest card Seven	
	2H 2D 4C 4D 4S	3C 3D 3S 9S 9D	
5	Full House	Full House	Player 1
	With Three Fours	with Three Threes	

The file, poker.txt, contains one-thousand random hands dealt to two players. Each line of the file contains ten cards (separated by a single space): the first five are Player 1's cards and the last five are Player 2's cards. You can assume that all hands are valid (no invalid characters or repeated cards), each player's hand is in no specific order, and in each hand there is a clear winner.

How many hands does Player 1 win?

2 Solution

```
import Data.List
import qualified Data.Map as Map
import Data.Maybe
import System.Environment
import Control.Monad

data Suit = Heart | Diamond | Club | Spade
  deriving (Eq, Ord, Show, Enum, Bounded)
data Rank = Two | Three | Four | Five | Six | Seven |
  Eight | Nine | Ten | Jack | Queen | King | Ace
  deriving (Eq, Ord, Show, Enum, Bounded)

data Card = Card
  {rank :: Rank
  , suit :: Suit}
  deriving Eq

instance Ord Card where
  compare x y
     | rank x ≡ rank y = compare (suit x) (suit y)
     | otherwise = compare (rank x) (rank y)

instance Show Card where
  show a = (show $ rank a) ++ " of " ++ (show $ suit a) ++ "s"

type DealtHand = [Card]

data Hand = RoyalFlush | StraitFlush | FourOfKind | FullHouse | Flush |
  Strait | ThreeOfKind | TwoPair | OnePair | HighCard
  deriving (Show, Ord, Enum, Eq, Bounded)

isFlush :: DealtHand → Bool
isFlush cds = all (λz → suit z ≡ suit (cds !! 0)) cds

isStrait :: DealtHand → Maybe Rank
isStrait cds =
  let nRanks = sort (map (fromEnum ∘ rank) cds)
      strt   = [(minimum nRanks) .. (maximum nRanks)]
  in if nRanks ≡ strt
     then Just $ (toEnum ∘ maximum) nRanks
     else Nothing

nOfAKind :: DealtHand → [(Int, Rank)]
nOfAKind cds =
  let nRanks = (group ∘ sort) (map rank cds)
      fours = filter (λz → length z ≡ 4) nRanks
      threes = filter (λz → length z ≡ 3) nRanks
      pairs = reverse $ sort $ filter (λz → length z ≡ 2) nRanks
      gps    = map (λz → (length z, head z)) (concat [fours, threes, pairs])
  in take 2 gps

hand1 cds =
  let ifl = isFlush cds
      str  = isStrait cds
      nknd = nOfAKind cds
      hnd1 = if ifl
        then case str of
          Just Ace → [(RoyalFlush, Nothing)]
          Just c → [(StraitFlush, Just c)]
          otherwise → [(Flush, Just $ maximum (map rank cds))]
        else case str of
```

3 Result

```
runhaskell problem54.lhs
```