Laboratorul 8 - Programare Logică și Funcțională

Seria 36

Mai 2025

1. a. Scrieţi o funcţie `removeOddHalfEven :: [Int] -> [Int]` care primeşte o listă de elemente numere întregi şi care elimină elementele impare, iar pe cele pare le înjumătăţeşte.
   b. Rescrieţi funcţia, folosind `map` şi `filter`.
   c. Rescrieţi funcţia, folosind comprehensiune.
2. a. Scrieţi o funcţie `multDigits :: String -> Int` care calculează produsul cifrelor dintr-un şir de caractere. Folosiţi `isDigit`, respectiv `digitToInt` din `Data.Char`.
   b. Rescrieţi funcţia, folosind `map` şi `filter`.
   c. Rescrieţi funcţia, folosind comprehensiune.
3. Scrieţi o funcţie recursivă care primeşte o listă de întregi şi returnează dublul elementelor impare, strict mai mici decât un `n` dat.
4. Scrieţi o funcţie care să calculeze suma elementelor pare de pe poziţii impare.
5. Scrieţi o funcţie care primeşte o listă de şiruri de caractere şi returnează o listă nouă de şiruri, care păstrează, din fiecare şir anterior, doar vocalele.
6. Rescrieţi exerciţiile 1-3 cu `foldr`, folosind *proprietatea de universalitate*.
7. Definiţi un tip de date

   ```
   data LList a = Nil | Cons a Nil
   ```

   unde o listă poate fi reprezentată:

   ```
   l1 :: LList Int
   l1 = Cons 1 $ Cons 2 $ Cons 3 Nil
   ```

   a. Scrieţi instanţe pentru `Eq` şi `Show` pentru `LList`.
   b. Definiţi o funcţie `lAppend :: LList a -> LList a -> LList a` care concatenează două liste.
   c. Definiţi instanţe pentru clasele `Semigroup` şi `Monoid` pentru `LList`.
   d. Implementaţi funcţiile `lFilter`, `lMap` şi `lFoldr` pentru tipul `LList`.
   d. Scrieţi o funcţie `lToList :: LList a -> [a]` care converteşte `LList a` în `[a]`.
8. Definiţi un tip de date

   ```
   data Point a b = Point a b
   ```

   a. Scrieţi instanţe pentru `Eq` şi `Show`, astfel încât reprezentarea ca `String` să fie de forma `(a, b)`.
   b. Definiţi o funcţie `lZip :: LList a -> LList b -> LList (Point a b)`.