

Simulare Examen PF (15 Aprilie 2025)

Universitatea Alexandru Ioan Cuza
Facultatea de Informatică
Programare Funcțională 2024-2025

Nume:

Grupa:

Problema 1 (8p)

Scrieți o funcție Haskell `patratPerfect` care primește la intrare un întreg strict pozitiv `n` și verifică dacă `n` este pătrat perfect. Soluția trebuie să se bazeze pe recursie. Puteți folosi funcții auxiliare.

.....

.....

.....

.....

.....

.....

.....

.....

.....

Problema 2 (8p)

Scrieți o funcție Haskell care calculează penultimul element al unei liste, fără a calcula inversul listei. Puteți presupune că lista are cel puțin două elemente. Soluția trebuie să se bazeze pe recursie. Puteți folosi funcții auxiliare, dar pentru cea mai simplă soluție nu aveți nevoie.

.....

.....

.....

.....

.....

.....

.....

.....

.....

Problema 3 (8p)

Fie următoarea funcție recursivă, care calculează numărul de elemente dintr-un arbore binar.

```
data Arb = Vid | Nod Int Arb Arb deriving (Show, Eq)
count :: Arb -> Int
count Vid = 0
count (Nod _ l r) = 1 + count l + count r
```

Scrieți o variantă *tail-recursive* a acestei funcții.
Indicație: folosiți o funcție auxiliară `countAux :: [Arb] -> Int -> Int` care calculează numărul de noduri ale tuturor arborilor din lista dată ca parametru.

.....

.....

.....

.....

.....

.....

.....

.....

.....

Problema 4 (8p)

Ce tip are expresia Haskell `filter (\x -> mod (x :: Int) 2 == 0) [1, 3, 4]`? Explicați.

Problema 5 (8p)

Folosind `map`, `filter`, și `foldr`, scrieți o expresie Haskell care calculează suma pătratelor numerelor pare dintr-o listă `l`.

```
map :: (a -> b) -> [a] -> [b]
filter :: (a -> Bool) -> [a] -> [a]
foldr :: (a -> b -> b) -> b -> [a] -> b
```

Problema 6 (8p)

Scriveți o funcție de ordin superior care are același comportament cu funcția `map` din biblioteca standard.

Problema 7 (8p)

Fie următorul tip de date, valorile căruia reprezintă numere naturale în format unar.

```
data Nat = Zero | Succ Nat deriving (Show, Eq)
```

Scrieți o funcție **quotientRemainder** care calculează câtul și restul împărțirii întregi a două astfel de numere. Puteți folosi funcții auxiliare. Indicație: cea mai simplă implementare este prin scăderi repetate.

.....
.....
.....

Problema 8 (8p)

Se dă următorul tip de date, care reprezintă expresii din aritmetica Presburger:

```
data Expr = Const Integer | Var String | Minus Expr | Plus Expr Expr | Mult Integer Expr
  deriving (Show, Eq)
```

Scrieți o funcție Haskell care transformă o expresie într-una echivalentă care nu folosește constructorul `Mult`.

.....
.....
.....
.....
.....
.....
.....
.....
.....

Problema 9 (8p)

Scrieți o funcție care primește ca parametru o listă de funcții booleene binare totale și testează dacă lista conține funcția *și logic*.

.....
.....
.....
.....
.....
.....
.....
.....
.....

Problema 10 (8p)

Instanțiați clasa `Show` cu tipul `Expr` definit la Problema 8 astfel încât expresia să apară în format infix.

```
data Expr = Const Integer | Var String | Minus Expr | Plus Expr Expr | Mult Integer Expr
  deriving Eq
```

```
instance Show Expr where
```

.....
.....
.....
.....
.....
.....
.....
.....
.....