Simulare Examen PF (Aprilie 2023)

Universitatea Alexandru Ioan Cuza
Facultatea de Informatică
Programare Funcțională 2022-2023
Nume: MOISĂ IULIA-ELENA
Grupa: ..A6...

Obiective de învățare.

O1 Capacitatea de a defini funcții recursive de dificultate medie (e.g.. funcții de sortare, funcții cunoscute de la matematică);

O2 Capacitatea de a proiecta și defini tipuri algebrice de date de dificultate medie (e.g., pentru reprezentarea unor expresii aritmetice sau logice);

O3 Capacitatea de a utiliza și de a înțelege funcții de ordin superior (e.g., map, filter, reduce) în contexte de dificultate medie;

O4 Capacitatea de a face calcule folosind regula de beta-reducere în lambda-calcul.

Subiecte examen

1 Problema 1 (Haskell, O1)

Scrieți o funcție Haskell **sumOfSquares** care primește la intrare un întreg strict pozitiv **n** și calculează suma pătratelor tuturor întregilor de la 1 la **n**. Soluția trebuie să se bazeze pe recursie.
Signatura funcției:

```
sumOfSquares :: Int -> Int
```

```
sumOfSquares 0 = 0
sumOfSquares 1 = 1
sumOfSquares n = n * n + sumOfSquares (n-1)
```

nu e necesar ✓

2 Problema 2 (Haskell, O1)

Scrieți o funcție Haskell **countOccurrences** care primește o listă de întregi, **xs**, și un întreg **n** ca input. Funcția numără câte apariții ale lui **n** se găsesc în **xs**. Soluția trebuie să se bazeze pe recursie structurală.
Signatura funcției:

```
countOccurrences :: Eq a => [a] -> a -> Int
```

```
countOccurances [] _ = 0
countOccurances (x:xs) b = let index = countOccurances xs b in
                           if (==) b x then index+1 else index
```

✓

3 Problema 3 (Haskell, O1)

Scrieți o funcție Haskell **longestIncreasingSubsequence** care primește o listă de întregi, **xs**. Funcția calculează lungimea celui mai lung subșir crescător. Subșirul conține elemente în ordine crescătoare, dar acestea nu sunt neapărat pe poziții consecutive.

Soluția trebuie să se bazeze pe recursie structurală. Nu aveți voie să folosiți biblioteci suplimentare, alta decât biblioteca standard Prelude.

Signatura funcției:

```
longestIncreasingSubsequence :: [Int] -> Int
```

```
longestIncreasingSubsequence [] = 0
longestIncreasingSubsequence [a] = 1
longestIncreasingSubsequence (x:xs) = let nrOfOcc = longestIncreasingSub-
                                      sequence xs in
                                      if (<=) x (head xs) then nrOfOcc+1 else nrOfOcc
```

nu calculează ce se cere

2 1 3

4 Problema 4 (Haskell, O2)

Proiectați un tip de date **Vehicle** care are 4 constructori: **Car**, **Ship**, **Bicycle** și **Truck**. Fiecare constructor reprezintă un tip de vehicul și veți include pentru fiecare vehicul câteva caracteristici, după cum urmează:

- mașinile vor avea asociate o marcă, un model și un an de fabricație;
- vapoarele vor avea asociate o marcă și un an de fabricație;
- bicicletele vor avea asociate o marcă și un model;
- camioanele vor avea asociate o marcă, un model și un tonaj.

```
data Vehicle = Car String String Int
             | Ship String Int
             | Bicycle String String
             | Truck String String Float
```

✓

5 Problema 5 (Haskell, O2)

Proiectați un tip de date care să permită cel puțin codificarea unor expresii aritmetice cum ar fi: **(x * 7 + 10) - 23**.

Scrieți expresia dată ca exemplu mai sus ca valoare Haskell care să aibă tipul pe care tocmai l-ați definit.

```
data Exp = Val Int
         | Var String
         | Add Exp Exp
         | Mul Exp Exp
         | Minus Exp Exp
         | Div Exp Exp
deriving Show
```

```
Div (Add (Mul (Var "x") (Val 7)) (Val 10)) (Val 2)
```

✓

6 Problema 6 (Haskell, O2)

Proiectați un tip de date **Shape** care să reprezinte diferite forme bidimensionale, cum ar fi cercuri, dreptunghiuri, sau triunghiuri. Cercul e determinat de rază, dreptunghiul de lățime și înălțime, triunghiul de cele trei laturi.

Definiți o funcție care calculează aria unei forme geometrice de tip **Shape**.

```
data Shape = Cerc Float
           | Dreptunghi Float Float
           | Patrat Float
aria :: Shape -> Float
aria (Cerc a) = pi * a * a
aria (Dreptunghi a b) = a * b
aria (Patrat a) = a * a
```

7 Problema 7 (Haskell, O3)

Scrieți o funcție Haskell **sumOfEvenSquares** care primește o listă de întregi la intrare și calculează suma pătratelor tuturor numerelor pare din listă.

Folosiți `map`, `foldr` sau `foldl`, și `filter` în soluție.

```
sumOfEvenSquares :: [Int] -> Int
```

De exemplu, `sumOfEvenSquares [1, 2, 3, 4, 5, 6, 7, 8]` trebuie să întoarcă 120.

```
sumOfEvenSquares = foldl (+) 0 . map (^2) . filter even
```

8 Problema 8 (Haskell, O3)

Scrieți o funcție de ordin superior **averageGrade** care calculează media notelor fiecărui student dintr-o listă. Folosiți `map` și (`foldr` sau `foldl`).

```
averageGrade :: [(String, [Float])] -> [(String, Float)]
```

```
averageGrade = map (\(name, grades) -> (name, foldr (+) 0 grades
               / fromIntegral (length grades)))
```

9 Problema 9 (Haskell, O3)

Proiectați o versiune type-safe a funcției **head** pentru liste, implementată folosind `foldr`.

```
myhead :: [a] -> Maybe a
```

```
myhead = foldr (\x _ -> Just x) Nothing
```