Laborator 14 (Subiecte Examen 2021-2022)

1 Problema 1 (Haskell, O1)

Scrieți o funcție recursivă `string2Integer :: String -> Integer` care primește la intrare un șir de caractere care conține doar cifre și convertește șirul într-un număr întreg. Nu este necesar să validați șirul de intrare.

Exemplu de utilizare: `convert "12234342" = 12234342`, `convert "" = 0`.

În implementare veți utiliza în mod netrivial funcția de mai jos:

```
char2Integer :: Char -> Integer
char2Integer c = (fromIntegral (ord c)) - 48
```

Puteți utiliza funcții ajutătoare cu acumulatori dacă este necesar. Cerința este ca măcar o funcție să fie recursivă și să nu utilizeze alte funcții din biblioteca standard în afară de cele aritmetice (de exemplu: `+`, `*`).

2 Problema 2 (Haskell, O1)

Scrieți o funcție recursivă `countCapitals :: String -> Int` care numără toate literele mari din alfabet care apar într-un șir de caractere dat ca input. Puteți utiliza funcții ajutătoare dacă este necesar. Cerința este ca măcar o funcție sa fie recursivă și să nu utilizeze alte funcții din biblioteca standard în afară de cele comune (de exemplu: `+`, `<=`).

3 Problema 3 (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;
- biciclete vor avea asociate o marcă și un model;
- camioanele vor avea asociate o marcă, un model și un tonaj.

Identificați cu atenție tipurile pentru fiecare dintre constructori. Scrieți o funcție `vehicleBrand :: Vehicle -> String` care returnează marca unui vehicul dat ca parametru.

4 Problema 4 (Haskell, O2)

Proiectați un tip de date care să permită cel putin 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.

5 Problema 5 (Haskell, O3)

Utilizati funcția `filter :: (a -> Bool) -> [a] -> [a]` și funcția `length :: [a] -> Int` (ambele predefinite în biblioteca standard) pentru implementarea unei funcții `countDigits :: String -> Int`, care numară câte cifre conține un șir de caractere.

6 Problema 6 (Haskell, O3)

Scrieți o funcție de nivel superior `appFOverList :: [a] -> (a -> [a]) -> [a]` care aplică fiecărui element al listei funcția dată ca parametru și concatenează rezultatele. Funcția ar trebui sa se comporte astfel:

```
ghci> appFOverList [10,100,0,-10] decrement
[9,99,-1,-11]
ghci> appFOverList [10,100] (\x -> [ x - 1, x + 1 ])
[9,11,99,101]
ghci> appFOverList [] decrement
[]
```

unde `decrement` este definită astfel:

```
decrement :: Int -> [Int]
decrement x = [x - 1]
```

7 Problema 7 (λ-calcul, O4)

Efectuați o β-reducere pornind de la termenul de mai jos $\left(\lambda x.\lambda y.x\ y\right)\ y$.
Atenție: folosiți substituția de tip *capture avoiding*.

8 Problema 8 (λ-calcul, O4)

Identificați cele trei *redex*-uri din termenul de mai jos și efectuați fiecare dintre cele 3 β-reduceri corespunzătoare:
$(\lambda x.x)\ \Big((\lambda y.y)\ (\lambda z.(\lambda x'.x')\ z)\Big)$.