Examen PP – Seria 2CC

11.06.2015

ATENȚIE: Aveți 2 ore . 10p per subiect . 100p necesare pentru nota maximă . **Justificați** răspunsurile!

---

1. Reduceti la forma normală expresia:
   $(\lambda y.((\lambda x.\lambda y.x\ \ y)\ \ \Omega)\ \ \lambda x.\lambda y.y)$
   Soluție:
   $(\underline{\lambda y}.((\lambda x.\lambda y.x\ \ y)\ \ \Omega)\ \ \underline{\lambda x.\lambda y.y})$
   $\rightarrow ((\underline{\lambda x}.\lambda y.x\ \ \underline{\lambda x.\lambda y.y})\ \ \Omega)$
   $\rightarrow (\underline{\lambda y}.\lambda x.\lambda y.y\ \ \underline{\Omega})$
   $\rightarrow \lambda x.\lambda y.y$

2. Scrieți o funcție setN în Racket care primește două liste L1 și L2 (fără duplicate) ca argumente și întoarce o listă care este intersecția celor două liste, luate ca mulțimi (rezultatul nu trebuie să conțină duplicate).
   Soluție:
   ```
    (define (setU L1 L2)
     (cond
       ((null?  L1) L2)
       ((member (car L1) L2) (setU (cdr L1) L2))
       (else (cons (car L1) (setU (cdr L1) L2)))
     ))
   ```
   sau
   ```
   (define (setU2 L1 L2) (foldr (λ (x L)
     (if (member x L) L (cons x L))) '() (append L1 L2)))
   ```

3. Date fiind funcțiile `E` și `F` și următorul cod care considerăm că se execută fără erori, de câte ori sunt evaluate fiecare dintre cele două funcții, și la ce linii din cod se fac evaluările?
   ```
   1.  (define fmic (λ (a)
   2.     (let [   (f (F a))   (g (delay (E a)))   ]
   3.      f )   ))
   4.  (fmic (E 'argument))
   ```
   Soluție:
   E la 4, F la 2

4. Sintetizați tipul funcției Haskell următoare: `f x = f (f x)`
   Soluție:
   ```
    f ::  a -> b
   x ::  a
   (f (f x)) ::  b
   ------
   f ::  c -> d
   (f x) ::  c
   d = b
   ------
   f ::  e -> g
   x ::  e = a
   g = c
   din f:  a = c = e și b = d = g
   f ::  a -> a
   ------
   f ::  t -> t
   ```

5. Instanțiați în Haskell clasa `Ord` pentru liste. Ordonarea listelor va fi dată de compararea primului element din fiecare listă. E.g. `[1, 2, 3] < [2, 3, 4]` pentru că $1 < 2$.
Soluție:
NOTĂ: Pentru ca implementarea să compileze am folosit aici `MyOrd` și `#<=` în loc de `Ord` și `<=`, definite astfel: `class Eq a => MyOrd a where (#<=) :: a -> a -> Bool`.
Soluția cerută, (dar cu `Ord` și `<=` în loc de `MyOrd` și `#<=`), era:
```
 instance Ord a => MyOrd [a] where
    (h1:_) #<= (h2:_) = h1 <= h2
```

6. Scrieți o funcție Haskell care elimină dintr-o listă elementele care apar de mai multe ori. E.g. `[1, 2, 3, 2, 3, 5] -> [1, 5]`.
Soluție:
```
 nodups [] = []
nodups l = head l :  nodups (filter (not .  (== head l)) $ tail l)
```

7. Care este fluxul `s` pentru care este adevărat:
`(take 10 $ zipWith (+) s [1, 1..])  == (take 10 $ tail s)`
Soluție:
Naturals

8. Traduceți în logica cu predicate de ordinul I următoarea propoziție:
Cine se scuză, se acuză.
Soluție:
$\forall x.scuza(x, x) \Rightarrow acuza(x, x)$

9. Știind că în mijlocul Dunării exista insula AdaKaleh, și există o trecere de pe malul drept pe insulă, și una de pe insulă pe malul stâng, iar trecerea este tranzitivă, folosiți rezoluția pentru a demonstra că se poate trece de pe malul drept pe malul stâng (folosiți predicatul $trece(loc, altLoc)$).
Soluție:
$trece(MStang, AdaKaleh)$ (1)
$trece(AdaKaleh, MDrept)$ (2)
$\forall x \forall y \forall z.trece(x, y) \wedge trece(y, z) \Rightarrow trece(x, z)$ (tranzitivitate)
    $\rightarrow \{\neg trece(x, y) \vee \neg trece(y, z) \vee trece(x, z)\}$ (3)
$\neg trece(MStang, MDrept)$ (4) (concluzie negată)
(1) rezolvă cu (3), substituție $\{MStang/x, AdaKaleh/y\}$
    $\rightarrow \neg trece(AdaKaleh, z) \vee trece(MStang, z)$ (5)
(2) rezolvă cu (5), substituție $\{MDrept/z\} \rightarrow trece(MStang, MDrept)$ (6)
(6) rezolvă cu (4) $\rightarrow$ clauza vidă.

10. Implementați în Prolog un predicat având semnătura `mapF(+L, -LO)`, care să fie echivalent cu expresia Haskell `(map f)`, știind că există deja definit un predicat `f(+X, -XO)`.
Soluție:
mapF(L, LO):- findall(XO, (member(X, L), f(X, XO)), LO).

11. Câte soluții are interogarea `p(L, [1, 2, 3])` în condițiile în care avem definiția de mai jos? Ce formă au aceste soluții?
`p(D, [A, B, C]) :- member(A, D), member(B, D), member(C, D).`
Soluție:
O infinitate. Soluțiile sunt liste care conțin 1, 2 și 3 (în ordine) și un număr ≥ 0 de elemente neinstanțiate.

12. Scrieți un algoritm Markov care lucrează pe un șir de simboluri din mulțimea A și înlocuiește fiecare grup de două simboluri identice cu o singură apariție a simbolului. Grupurile identificate trebuie să fie disjuncte. De exemplu, pentru șirul `0100110110001` se obține `010101001`.

    Soluție:

    `Dedup(); Ab` g_1
      `a`$g_1 g_1$`->` g_1`a`
      `a`g_1`->` g_1`a`
      `a -> .`
      `-> a`