# 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:
   $(((\lambda x.\lambda y.\lambda z.y\ \ \lambda x.x)\ \ (\lambda z.\lambda t.z\ \ z))\ \ \Omega)$
   *Soluție:*
   $(((\lambda x.\lambda y.\lambda z.y\ \ \lambda x.x)\ \ (\lambda z.\lambda t.z\ \ z))\ \ \Omega)$
   $\rightarrow (((\underline{\lambda x}.\lambda y.\lambda z.y\ \ \underline{\lambda x.x})\ \ (\lambda z.\lambda t.z\ \ z))\ \ \Omega)$
   $\rightarrow ((\underline{\lambda y}.\lambda z.y\ \ \underline{(\lambda z.\lambda t.z\ \ z)})\ \ \Omega)$
   $\rightarrow (\underline{\lambda \mathbf{w}}.(\lambda z.\lambda t.z\ \ z)\ \ \underline{\Omega})$
   $\rightarrow (\underline{\lambda \mathbf{v}}.\lambda t.\mathbf{v}\ \ \underline{z})$
   $\rightarrow \lambda t.z$

2. Scrieți o funcție setU în Racket care primește două liste L1 și L2 (fără duplicate) ca argumente și întoarce o listă care este reuniunea celor două liste, luate ca mulțimi (rezultatul nu trebuie să conțină duplicate).
   *Soluție:*
   ```
   (define (setN L1 L2)
    (cond
      ((null?  L1) '())
      ((member (car L1) L2) (cons (car L1) (setN (cdr L1) L2)))
      (else (setN (cdr L1) L2))
    ))
   ```
   *sau*
   ```
   (define (setN2 L1 L2) (filter (λ (x) (member x L2)) L1))
   ```

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 gmic (λ (a)
   2.     (let [   (f (delay (F a)))   (x (g a))   ]
   3.     f )   ))
   4.  (gmic (E 'argument))
   ```
   *Soluție:*
   E la 4, F niciodată.

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

5. Instanțiați în Haskell clasa `Ord` pentru perechi. Ordinea perechilor va fi dată de compararea celui de-al doilea element din pereche. E.g. `(1, 2) > (2, 0)` (pentru că $2 > 0$).
*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 (Eq a, Ord b) => MyOrd (a, b) where
  (_, y1) #<= (_, y2) = y1 <= y2
```

6. Scrieți o funcție Haskell care păstrează dintr-o listă doar valorile care apar de mai multe ori. E.g. `[1, 2, 3, 2, 3] -> [2, 3]`.
*Soluție:*
```
dups [] = []
dups (h:t)
  | elem h t = h :  dups (filter (/= h) t)
  | otherwise = dups t
```

7. Care este fluxul `s` pentru care este adevărat:
`(take 10 $ zipWith (+) s (tail s)) == (take 10 $ (tail .  tail) s)`
*Soluție:*
Fibonacci

8. Traduceți în logica cu predicate de ordinul I următoarea propoziție:
*Cine are carte, are parte.*
*Soluție:*
$\forall x.are(x, Carte) \Rightarrow are(x, Parte)$

9. Știind că elefantul este mai mare decât leul, și leul este mai mare decât șoricelul, iar relația de 'mai mare' este tranzitivă, folosiți rezoluția pentru a demonstra că elefantul este mai mare decât șoricelul
*Soluție:*
$maiMare(Elefant, Leu)$ (1)
$maiMare(Leu, Soricel)$ (2)
$\forall x \forall y \forall z.maiMare(x, y) \wedge maiMare(y, z) \Rightarrow maiMare(x, z)$ (tranzitivitate)
$\rightarrow \{\neg maiMare(x, y) \vee \neg maiMare(y, z) \vee maiMare(x, z)\}$ (3)
$\neg maiMare(Elefant, Soricel)$ (4) (concluzie negată)
(1) rezolvă cu (3), substituție $\{Elefant/x, Leu/y\}$
$\rightarrow \neg maiMare(Leu, z) \vee maiMare(Elefant, z)$ (5)
(2) rezolvă cu (5), substituție $\{Soricel/z\} \rightarrow maiMare(Elefant, z)$ (6)
(6) rezolvă cu (4) $\rightarrow$ clauza vidă.

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

11. Câte soluții are interogarea `p([1, 2, 3], L)` 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:*
27. Sunt toate combinațiile de 1,2,3, inclusiv în care un element apare de mai multe ori.

12. Scrieți un algoritm Markov care lucrează pe un șir de simboluri din mulțimea A și înlocuiește fiecare grup de două sau mai multe simboluri identice consecutive cu o singură apariție a simbolului. De exemplu, pentru șirul `0100110110001` se obține `01010101`.
*Soluție:*
    1. `Dedup(); Ab` $g_1$
    2. $g_1g_1$`->` $g_1$
    3. `->  .`

<