

Examen PP – Seria 2CC

11.06.2016

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

1. Ilustrați cele două posibile secvențe de reducere pentru expresia: $(\lambda y.(\lambda x.\lambda y.x \ y) \ 2)$
2. Implementați în Racket o funcție `myAndMap` care să aibă un comportament similar cu `andmap` – primește o listă și întoarce o valoare booleană egală cu rezultatul operației `and` pe elementele listei. Folosiți cel puțin o funcțională. Nu folosiți `andmap`.
3. Ce întoarce următoarea expresie în Racket? Justificați!

```
(let ((n 2))  
  (letrec ((f (lambda (n)  
               (if (zero? n) 1 (* n (f (- n 1)))))))  
    (f 5))  
  )
```
4. Cum se poate îmbunătăți următorul cod Racket pentru ca funcția `calcul-complex` să se evalueze doar atunci când este necesar, adică doar atunci când `variant` este fals (fără a o muta apelul lui `calcul-complex` în interiorul lui `calcul`) ?
 1.

```
(define (calcul x y z) (if x y z))
```
 2.

```
(define (test variant) (calcul variant 2 (calcul-complex 3)))
```
5. Sintetizați tipul funcției `f` în Haskell: $f \ x \ y = (y \ x) \ x$
6. Instanțiați în Haskell clasa `Eq` pentru tripluri, considerând că $(a1, a2, a3)$ este egal cu $(b1, b2, b3)$ dacă $a1 == b1$ și $a2 == b2$.
7. Implementați în Haskell, fără a utiliza recursivitate explicită, funcția `setD` care realizează diferența a două mulțimi `a` și `b` ($a \setminus b$) date ca liste (fără duplicate). Care este tipul funcției?
8. Traduceți în logica cu predicate de ordinul întâi propoziția: *Orice naș își are nașul*.
9. Știind că $\forall x.Trezit(x, Dimineata) \Rightarrow \forall y.AjungeLa(x, y)$ și că $Trezit(Eu, Dimineata)$, demonstrați, folosind **metoda rezoluției**, că $AjungeLa(Eu, Examen)$.
10. Care este efectul aplicării predicatului `p` asupra listelor `L1` și `L2` (la ce este legat argumentul `R` în apelul `p(L1, L2, R)`)?
 $p(A, [], A). \ p(A, [E|T], [E|R]) :- p(A, T, R).$
11. Implementați un algoritm Markov care primește un șir de simboluri 0 și 1 și verifică dacă șirul începe cu 0 și se termină cu 1 și, în caz afirmativ, adaugă la sfârșitul șirului simbolurile "ok", altfel nu schimbă șirul cu nimic. Exemple: $010111011 \rightarrow 010111011ok$; $010 \rightarrow 010$; $1010 \rightarrow 1010$
12. Explicați care dintre următoarele apeluri dă eroare și care nu, și justificați pentru fiecare:
 1.

```
(if #t 5 (/ 2 0))
```

 (Racket)
 2.

```
(let ((f (lambda (x y) x))) (f 5 (/ 2 0)))
```

 (Racket)
 3.

```
let f x y = x in f 5 (div 2 0)
```

 (Haskell)
 4. $X = 2 / 0, Y = X.$ (Prolog)