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: $(\lambda x.(\lambda y.\lambda x.y\ \ x)\ \ 5)$

2. Implementați în Racket o funcție `myOrMap` care să aibă un comportament similar cu `ormap` – primește o listă și întoarce o valoare booleană egală cu rezultatul operației `or` pe elementele listei. Folosiți cel puțin o funcțională. Nu folosiți `ormap`.

3. Ce întoarce următoarea expresie în Racket? Justificați!
```
(letrec ((f (lambda (n)
  (let ((n (- n 1)))
    (if (eq?  n -1) 1 (* (+ n 1) (f n)))))))
  (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 = x (y x)`

6. Instanțiați în Haskell clasa `Ord` pentru tripluri (știind că `Eq` este deja instanțiată), considerând că `(a1, a2, a3)` este mai mic decât `(b1, b2, b3)` dacă `a1 < b1`.

7. Implementați în Haskell, fără a utiliza recursivitate explicită, funcția `setN` care realizează intersecția a două mulțimi `a` și `b` date ca liste (fără duplicate). Care este tipul funcției?

8. Traduceți în logica cu predicate de ordinul întâi propoziția: *Orice copil are o mamă.*

9. Știind că $\forall x.Are(x, Carte) \Rightarrow \forall y.Are(x, y)$ și că $Are(Eu, Carte)$, demonstrați, folosind **metoda rezoluției**, că $Are(Eu, Parte)$.

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([E|T], A, [E|R]) :- p(T, A, R).
```

11. Implementați un algoritm Markov care primește un șir de simboluri `0` și `1` și verifică dacă șirul începe cu `1` și se termină cu `0` și, în caz afirmativ, adaugă la sfârșitul șirului simbolurile "`ok`", altfel nu schimbă șirul cu nimic. Exemple: `1110100` → `1110100ok` ; `0101` → `0101` ; `010` → `010` ; `1010` → `1010ok`

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 (λ (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)

<