

# Examen PP – Seria CC — NOT EXAM MODE

16.06.2017

Timp de lucru 2 ore . 100p necesare pentru nota maximă

1. Determinați forma normală pentru următoarea expresie, ilustrând pașii de reducere:

$((\lambda x.\lambda y.\lambda z.(x\ y)\ \lambda x.y)\ a)$

*Soluție:*

$((\lambda x.\lambda y.\lambda z.(x\ y)\ \lambda x.y)\ a) \rightarrow_{\alpha} ((\lambda x.\lambda w.\lambda z.(x\ w)\ \lambda x.y)\ a) \rightarrow_{\beta} (\lambda w.\lambda z.(\lambda x.y\ w)\ a) \rightarrow_{\beta} \lambda z.(\lambda x.y\ a) \rightarrow_{\beta} \lambda z.y$

2. Este vreo diferență (ca efect, la execuție) între cele două linii de cod Racket? Dacă da, care este diferența?; dacă nu, de ce nu diferă?

```
(define a 2) (let ((c 2)) (let ((a 1) (b a)) (+ a b)))
```

```
(define a 2) (let* ((c 2) (a 1) (b a)) (+ a b))
```

*Soluție:*

În prima linie, definiția (a 1) este vizibilă în corpul let-ului, dar nu și în definiția lui b, care vede încă a=2; prima linie dă 3, a doua dă 2.

3. Implementați în Racket funcția f care primește o listă și determină elementul cu cel mai mare modul. Folosiți, în mod obligatoriu, cel puțin o funcțională.

*Soluție:*

```
(car (filter (lambda (e) (null? (filter (compose ((curry <) (abs e)) abs) L))) L))
```

*sau*

```
(car (filter (lambda (e) (null? (filter (lambda (a) (< (abs e) (abs a))) L))) L))
```

*sau*

```
(let ((M (last (sort (map abs L) <)))) (if (member M L) M (- 0 M)))
```

4. Sintetizați tipul funcției f (în Haskell):  $f\ x\ y\ z\ g = \text{map } g\ [x, y, z]$

*Soluție:*

$f :: a \rightarrow b \rightarrow c \rightarrow d \rightarrow e$

$d = g1 \rightarrow g2$

$\text{map} :: (t1 \rightarrow t2) \rightarrow [t1] \rightarrow [t2]$

$a = b = c$  (parte din aceeași listă)

$t1 = a = b = c$

$e = [t2]$

$f :: t1 \rightarrow t1 \rightarrow t1 \rightarrow (t1 \rightarrow t2) \rightarrow [t2]$

5. Scrieți definiția în Haskell a clasei Ended care, pentru un tip colecție t construit peste un alt tip v, definește o funcție frontEnd care extrage primul element din colecție și o funcție backEnd care extrage ultimul element din colecție.

Instanțiați această clasă pentru tipul data NestedL a = A a | L [NestedL a]

*Soluție:*

```
class Ended t where frontEnd :: t v -> v; backEnd :: t v -> v
```

```
instance Ended NestedL where
```

```
    frontEnd (A a) = a; frontEnd (L l) = frontEnd $ head l
```

```
    backEnd (A a) = a; backEnd (L l) = backEnd $ last l
```

6. Știind că *Un bogat când moare, săracul fluiere*, și că *bogat(Bill), sarac(Bob), și moare(Bill)*, demonstrați folosind rezoluția că *fluiere(Bob)* este adevărat .

*Soluție:*

$\forall x.\forall y.\text{bogat}(x) \wedge \text{sarac}(y) \wedge \text{moare}(x) \rightarrow \text{fluiere}(y)$

$\neg\text{bogat}(x) \vee \neg\text{sarac}(y) \vee \neg\text{moare}(x) \vee \text{fluiere}(y)$

$+bogat(Bill)\{x \leftarrow Bill\} \Rightarrow$   
 $\neg sarac(y) \vee \neg moare(Bill) \vee fluiera(y)$   
 $+ \neg fluiera(Bob)\{ \Rightarrow \neg sarac(Bob) \vee \neg moare(Bill)$   
 $+ moare(Bill) \Rightarrow \neg sarac(Bob)$   
 $+ sarac(Bob) \Rightarrow$  clauza vidă

7. Implementați în Prolog predicatul  $x(L, M)$  care detemină, pentru o listă  $L$ ,  $M$ , maximul listei. Nu folosiți recursivitate explicită.

*Soluție:*

$x(L, M) :- member(M, L), forall(member(E, L), X > E).$

8. Implementați un algorim Markov care primește în șirul de intrare un număr binar și adună 2 la acest număr. Exemple:  $1 + 10 = 11$ ;  $10 + 10 = 100$ ;  $1000 + 10 = 1010$ ;  $101 + 10 = 111$ ;  $111 + 10 = 1001$ .

*Soluție:*

$ag \rightarrow ga$   
 $ga \rightarrow bg$   
 $0b \rightarrow 1c$   
 $1b \rightarrow b0$   
 $b \rightarrow 1c$   
 $c \rightarrow .$   
 $\rightarrow a$