Precizări:

- Fiecare subiect are câte 10p. **Punctajul NU se acordă în absența justificării răspunsului!**
- Este suficientă rezolvarea a **10** subiecte din cele 12 pentru nota maximă.
- Punctajul suplimentar reprezintă **bonus**, ce poate compensa punctajul de pe parcurs.

1. Fie următoarea reprezentare a numerelor naturale în calculul lambda: $0 \equiv \lambda x.x$, $1 \equiv \lambda x.\lambda x.x$, $2 \equiv \lambda x.\lambda x.\lambda x.x$ etc. Scrieți definiția funcției de incrementare, *inc*, astfel încât $(inc\ \ 0) \to 1$, $(inc\ \ 1) \to 2$ etc.

   Soluție. .

   $inc \equiv \lambda n.\lambda x.n.$ □

2. Fie următoarea funcție în Racket:

```
1 (define (f x)
2   (lambda (y)
3     (+ (* x 3) y)))
```

   Cum decurge evaluarea aplicației `(f (+ 1 2))`, utilizând modelele bazate pe

   (a) (5p) substituție textuală

   (b) (5p) evaluare contextuală?

   Soluție. .

   Ambele situații presupun mai întâi evaluarea parametrului la 3.

   (a) $\lambda y.(+\ \ (*\ \ 3\ \ 3)\ \ y)$

   (b) $\langle \lambda y.(+\ \ (*\ \ x\ \ 3)\ \ y);\ \{x \leftarrow 3\} \rangle$ □

3. Fie următoarea funcție în Racket:

```
1 (define (f x)
2   (lambda (y)
3     (if (eq? y 'halt)
4         x
5         (f (+ x y)))))
```

   (a) (5p) Ce face `f`?

   (b) (5p) Dați un exemplu de expresie care surprinde utilitatea lui `f`.

   Soluție. .

   (a) `f` calculează suma parametrilor primiți până la întâlnirea simbolului `'halt`.

   (b) `((((f 1) 2) 3) 'halt)` → `6` □

4. Fie următoarea reprezentare în Haskell a unui arbore binar:

```haskell
data Tree a
    = Empty
    | Node (Tree a) a (Tree a)
```

Implementați pentru tipul `Tree a` o funcțională analoagă lui `foldr`, care să acumuleze toate cheile din arbore în ordinea subarbore drept – rădăcină – subarbore stâng, pornind de la definițiile de mai jos:

```haskell
foldr :: (a -> b -> b) -> b -> [a] -> b
foldr f acc [] = acc
foldr f acc (h : t) = f h (foldr f acc t)

foldTree :: (a -> b -> b) -> b -> Tree a -> b
...
```

Hint: O listă poate fi privită ca un arbore în care niciun nod nu are subarbore stâng.

Soluție. .

```haskell
foldT _ acc Empty = acc
foldT f acc (Node left key right) =
    foldT f (f key (foldT f acc right)) left
```
□

5. Care sunt valorile expresiilor din liniile 7 și 10 în urma execuției următoarei secvențe în Racket (presupunând utilizarea unui limbaj care permite `define`-uri multiple)?

```racket
(define f
  (let ([x 1])
    (lambda (y)
      (+ x y))))

(define x 0)
(let ([x 10]) (f 0))

(define x 10)
(let ([x 0]) (f 0))
```

Soluție. .

Ambele valori sunt 1, pentru că apariția lui `x` din linia 4 este obținută static pe baza definiției din linia 2. □

6. Fie următoarea definiție și aplicare în linia de comandă a unei funcții în Haskell:

```haskell
f x y = x + x
> f (1 + 2) (3 + 4)
```

Rescrieți cele două expresii în Racket, astfel încât să obțineți aceleași efecte ca în Haskell, în privința modalității de evaluare.

Soluție. .

```
(define (f x y)
  (+ (force x) (force x)))

> (f (delay (+ 1 2)) (delay (+ 3 4)))
```
□

7. Sintetizați tipul următoarei funcții în Haskell:

```
f x y z = foldr (.) id [x, y, z]
```

Tipul operatorului de compunere este

```
(.) :: (d -> e) -> (c -> d) -> c -> e
```

Soluție. .

```
f :: g -> h -> i -> j
foldr :: (a -> b -> b) -> b -> [a] -> b
a = d -> e  -- foldr + (.)
b = c -> d = c -> e  -- foldr + (.)
b = k -> k  -- foldr + id
c = d = e = k
g = h = i = a  -- foldr + lista
f :: (k -> k) -> (k -> k) -> (k -> k) -> k -> k
```
□

8. Supraîncărcați în Haskell operatorul de adunare pentru funcții unare. Într-un punct oarecare, valoarea funției rezultate se obține prin însumarea valorilor funcțiilor în punctul respectiv.

Soluție. .

```
instance Num b => Num (a -> b) where
    (f + g) x = f x + g x
    -- sau: f + g = \x -> f x + g x
```
□

9. Fie propozițiile din logica cu predicate de ordinul I:

$$\forall x.\exists y.(x \leq y) \tag{1}$$
$$\exists y.\forall x.(x \leq y) \tag{2}$$

(a) (5p) Sub o interpretare în care domeniul este reprezentat de mulțimea numerelor naturale, care dintre cele două propoziții este adevărată?

(b) (5p) Cum trebuie modificat domeniul pentru ca ambele propoziții să fie adevărate?

Soluție. .

(a) Doar propoziția (1) este adevărată, deoarece orice număr natural este mai mic sau egal cu sine ($y = x$), însă nu putem găsi un cel mai mare număr natural, așa cum pretinde propoziția (2).

(b) Domeniul trebuie să fie finit. □

10. Utilizând metoda rezoluției, demonstrați că (a este o constantă):

$$\{\forall x.P(x) \Rightarrow Q(x), P(a)\} \models \exists x.Q(x).$$

Soluție. .

Utilizăm reducerea la absurd, prin care adăugăm negația concluziei la mulțimea de propoziții: $\neg\exists x.Q(x) \equiv \forall x.\neg Q(x)$. Astfel, în urma aplicării procedurii de transformare în forma normală conjunctivă, care presupune și redenumirea unor variabile, se obțin clauzele $\{\neg P(x), Q(x)\}, \{P(a)\}, \{\neg Q(y)\}$. Aplicând rezoluția în raport cu substituția $\{y \leftarrow x, x \leftarrow a\}$, se obține clauza vidă. □

11. Transcrieți ca o **unică** propoziție logică următorul program Prolog. Punctajul **NU** se acordă pentru mai mult de o propoziție echivalentă.

```
1 c :- a.  c :- b.
2 d :- a.  d :- b.
```

Soluție. .

$a \vee b \Rightarrow c \wedge d.$ □

12. Scrieți un algortim Markov care să elimine toate copiile **adiacente** ale unui simbol de pe bandă, păstrând un singur exemplar. Spre exemplu, dacă șirul inițial este `122233112`, se va obține în final `12312`.

Soluție. .

```
1 Clean()
2     gg -> g
3     -> .
4 end
```
□