Nume: _____

1	2	3	4	5	6	7	8	9	10	10	10

Examen PP CC – var B 03.06.2025

ATENȚIE: Aveți 2 ore · 1-9: 10p; 10: 30p · 100p pentru nota maximă · **Justificați** răspunsurile!

---

1. (a) Reduceți la forma normală expresia de mai jos, ilustrând pașii de reducere sau justificând cu rigurozitate. (b) Ce variabile libere și ce variabile legate există în expresie?
   $((\lambda x.\lambda y.x\ \ ((\lambda x.\lambda y.y\ \ A)\ \ B))\ \ ((\lambda x.\lambda y.y\ \ C)\ \ D))$
2. Descrieți care este efectul următorului program Racket, și a evaluării expresiei Haskell de mai jos:
   1. `(define (A) (A)) (define (B) (A)) (define C 1) (define D 2)`
   2. `([lambda (x y) x] ([lambda (x y) y] (A) (B)) ([lambda (x y) y] C D))`

   ```
   let a = a; b = b; c = 1; d = 2 in (\x y -> x) ((\x y -> y) a b) ((\x y -> y) c d)
   ```
3. La ce se evaluează următoarea expresie în Racket?
   `(let* [(x 3) (y 4) (f (delay (λ (y) (+ x y))))]  (let [(x 1)] ((force f) x)))`
4. Construiți în Haskell fluxul x în care $x_0 = 2$, $x_1 = 0$, și $x_n = \sum_{i=0}^{n-2}\{x_i \text{ dacă } x_i \text{ par altfel } 0\}$, pentru $n \geq 2$. Fluxul este 2,0,2,2,4,6,10,16,26,42,68,110,... și, de exemplu, elementul 16, de pe poziția 7, este suma elementelor pare de pe pozițiile 0-5. **Evitați recursivitatea explicită**.
5. Sintetizați, **ilustrând** procesul de sinteză, tipul funcției Haskell `zipWith (.) . map t`
6. Instanțiați în Haskell clasa `Num` pentru triplete de valori (fiecare valoare de un tip arbitrar), implementând numai operatorul `+`, care să efecuteze operația cerută numai pe prima valoare din fiecare triplet, în rest păstrând valorile din primul triplet.
7. Știind că "Pe cine nu doare, nu câștigă", demonstrați prin metoda rezoluției că dacă *câștigă*(Ion), atunci *doare*(Ion).
8. Scrieți un predicat Prolog `up` care concatenează secvențele (cel puțin două elemente) strict crescătoare dintr-o listă. Exemplu:
   `up([5, 1, 2, 3, 2, 3, 1, 1, 0, 9, 10], LS)` → `LS = [1, 2, 3, 2, 3, 0, 9, 10]`
9. Implementați în Prolog, folosind metapredicate, predicatul `getBest(+List, -BestValue)`, care primește în `List` o listă de perechi cheie – valoare, și, dat fiind un predicat `cond`, întoarce *cheia* corespunzătoare celei mai mici *valori* dintre perechile cu o *valoare* care respectă `cond`.
10. Implementați cerințele următoare într-un limbaj la alegere dintre Haskell, Racket și Prolog, **evitând recursivitatea explicită**. Considerăm o serie de timp ca fiind o serie de valori reale, asociate cu momente de timp (valori întregi). De exemplu, putem avea seria `[20, 15, 16, 19, 18, 20, 18, 22, 22, 21, 20, 25]`, cu valori pentru 12 momente de timp. Putem *eticheta* fiecare valoare din seria de timp cu "unknown", "up" (mai mare decât precedenta), "down" (mai mică) sau "same" (la fel).
    (a) În Haskell, definiți tipul (tipurile) de date, ca tip(uri) noi, necesare pentru a reprezenta o serie de timp, inclusiv cu suport pentru etichete. În celelalte limbaje, descrieți modalitatea de reprezentare. Implementați funcția/predicatul `create` care pentru o valoare `start` și o valoare `stop` construiește seria de timp cu valorile dintre momentele `start` și `stop`, cu etichete "unknown". Pentru a obține valoarea de la fiecare moment de timp, folosiți o funcție / un predicat `get` care primește un moment de timp și întoarce valoarea de la acel moment de timp. Considerăm `get` deja definit(ă).
    (b) Implementați funcția/predicatul `label` care primește o serie de timp și întoarce o serie de timp cu etichete asociate cu fiecare valoare. Pentru primul element al seriei de tip se va asocia eticheta "unknown". Pentru exemplul de mai sus și seria de timp între momentele 2 și 6 va rezulta o serie
    `2 - 16.0 - Unknown, 3 - 19.0 - Up, 4 - 18.0 - Down, 5 - 20.0 - Up, 6 - 18.0 - Down`
    (c) Implementați funcțiile/predicatele `select`, care primește o etichetă și o serie de timp și selectează prima valoare și apoi valorile cu eticheta dată; și `deltas`, care primește o serie de timp și calculează diferențele dintre valori succesive. Petnru seria de la (b), `select` cu eticheta "up" va da seria de tip `2 - 16.0 - Unknown, 3 - 19.0 - Up, 5 - 20.0 - Up`, iar `deltas` peste această valoare va da lista `[3.0,1.0]`.