Examen PP/2CD/28.05.19 Timp: 2h Ex 1-9: câte 10p, Pb 10: 30p, pentru maxim sunt suficiente 100p/120p

Numele și grupa	1	2	3	4	5	6	7	8	9	10

A

1. Încercuiți aparițiile legate și subliniați aparițiile libere ale variabilelor din expresia `(λx.(λy.(x y) λz.(y z)) x)`.

2. Pentru codul Racket de mai jos, scrieți evoluția pas cu pas a execuției lui `(pow 2 5)`, și ilustrați stiva la fiecare pas.

```
(define (pow a n)
  (cond ((zero? n) 1)
        ((odd? n)  (* a (pow a (- n 1))))
        (else      (pow (* a a) (/ n 2)))))
```

3. În Racket, fără a folosi recursivitate explicită:

   a) Scrieți o funcție curry care primește un predicat p și o listă L și întoarce lista elementelor x din L pentru care x respectă și x/2 (împărțire întreagă) nu respectă p.

   b) Folosiți funcția de mai sus, scriind minimul necesar, pentru a extrage elementele divizibile cu 2 dar nu cu 4 ale unei liste numbers.

4. Folosind interfața Racket pentru fluxuri, implementați fluxul (1), (1 1), (1 1), (1 1 1), (1 1 1), (1 1 1 1), (1 1 1 1), … (primul element o dată, apoi fiecare element de câte 2 ori). Apoi implementați același flux folosind promisiuni, ca și cum interfața nu ar exista.

5. Sintetizați tipul funcției f (în Haskell): f x y = iterate $ x . y

6. Folosind un list comprehension (5p), instanțiați clasa Ord pentru tipul Student de mai jos (fiecare student este definit prin nume și o listă de triplete de forma (nume_materie, notă_materie, credite_materie)) astfel încât studenții să fie ordonați după totalul creditelor la materii la care au obținut notă de trecere.

```
data Student = Student String [(String, Int, Int)] deriving (Eq, Show)
```

7. Traduceți în FOL propoziția: “Orice mâță blândă ori doarme, ori îl zgârie pe vreun om rău.”

8. Adăugați implementarea predicatului one_bubble (care va trece o dată prin listă interschimbând elementele vecine care sunt în ordine strict descrescătoare) la următorul cod Prolog pentru algoritmul bubble-sort:

```
bubble_sort(L, L) :- one_bubble(L, L), !.
bubble_sort(L, S) :- one_bubble(L, Bubbled), bubble_sort(Bubbled, S).
```

9. Pentru implementarea de mai jos, spuneți în câte moduri (și care sunt acestea, din punct de vedere al legării variabilelor) se va satisface scopul **sel(Y, [X, 2, Y, 4], 2, [1, Z, Z, 4]).**

```
sel(X, [X|List], Y, [Y|List]).
sel(X, [X0|XList], Y, [X0|YList]) :- sel(X, XList, Y, YList).
```

10. Problemă (de rezolvat în Haskell)

    a) Implementați un tip de date MList pentru liste ce pot conține întregi, caractere sau perechi de întregi și caractere.

    b) Implementați o funcție: filter' :: Char -> MList -> MList care filtrează o MListă astfel:
    Dacă primul parametru are valoarea:
    - 'i', atunci filter' întoarce o MListă ce conține doar valorile întregi
    - 'c', atunci filter' întoarce o MListă ce conține doar caracterele
    - 'p', atunci filter' întoarce o MListă ce conține doar perechile

    c) Implementați o funcție care primește o MListă și întoarce lista caracterelor conținute, dacă în MListă se află doar caractere, sau o valoare cu semnificație de eșec - altfel. Valoarea cu semnificație de eșec nu trebuie să se poată obține dintr-o MListă care conține doar caractere (de exemplu, [ ] nu indică un eșec cert, ci ar putea proveni dintr-o MListă goală).