1. Ce rezultat produce urmatorul cod Scheme? (2p) Argumentati pe scurt. (8p)

```
(define x 10)
(define y (delay (lambda() (/ x 2))))
(define (z) ((force y))) ; (define z (lambda () ((force y)))...)
(define t (z))
(define x 100)
(list t z (z))
```

(5 #<procedure:z> 50)
z – este definita ca o functie de zero argumente (define (z) …), prin urmare se evalueaza la #<procedure:z>
t - este definita ca ((force (delay (lambda () (/ x 2))))) = (/ 10 2) = 5, intrucat la momentul definirii lui t x=10
(z) – provoaca evaluarea ((lambda () (/ x 2))), intrucat expresia a fost deja fortata iar (lambda () (/ x 2)) a fost rezultatul salvat in cache; in contextul evaluarii acestei expresii, x este 100, deci rezultatul evaluarii este 50

2. Ce inseamna functii stricte/nestricte? (2p) Cum sunt functiile din Scheme din acest punct de vedere? Sunt toate la fel? (2p) Dati exemple. (3p) Dati un exemplu de cod care s-ar comporta diferit daca o functie ar fi stricta/nestricta. (3p)

Functii stricte = isi evalueaza argumentele la apel
Functii nestricte = nu isi evalueaza argumentele la apel
Functiile Scheme sunt in general stricte (+, -, functiile definite de utilizator cu define…), cu cateva exceptii (if, and, or…).
(if (< 2 3) 2 (/ 1 0))
- intoarce 2 daca functia if e nestricta
- da eroare de tip “impartire la 0” daca functia if e stricta

3. Precizati ce calculeaza urmatoarea functie Scheme. (2p) Rescrieti-o folosind functionale. (8p)

```
(define (f L)
  (if (null? L)
      0
      (if (zero? (modulo (car L) 2))
          (add1 (f (cdr L)))
          (f (cdr L)))))
```

---

f – calculeaza numarul de elemente pare din lista L
(define (f L) (length (filter even? L))))

4. Ce inseamna functii curry/uncurry? (2p) Dati un exemplu in care optiunea pentru curry/uncurry va ajuta sa va definiti in mod compact un numar de 3 functii similare. (8p)

Functii uncurry = la apel, isi primesc obligatoriu toate argumentele deodata

Functii curry = isi pot primi doar o parte din argumente la apel; rezultatul produs este o noua functie care asteapta restul de argumente

```
(define (power n)
  (lambda (x)
    (expt x n)))

(define id (power 1))
(define square (power 2))
(define cube (power 3))
(id 5) ;5
(square 5) ;25
(cube 5) ;125
```

5. Ce reprezinta principiul refractiei? (2p) Precizati o bucata de cod care ar rula diferit daca nu s-ar aplica acest principiu. (8p)

Principiul refractiei = o regula se aplica o singura data pe un acelasi set de fapte cu un acelasi set de legari ale variabilelor din patternuri la valori.

```
(defrule print-like-crazy
=>
   (printout t "I was gonna stop this print but then I got high" crlf))
```

6. Scrieti un program CLIPS care determina elementul cu numar maxim de aparitii intr-o lista (exemplu: (lista a 2 4 2 c c 4 c) => (rezultat c)). (10p)

```
(deffacts facts (lista a 2 4 2 c c 4 c))

(defrule init
        (lista $? ?x $?)
        (not (occ ?x ?))
  =>
        (assert (occ ?x 0)))

(defrule count
        ?fl <- (lista ?x $?s)
        ?fo <- (occ ?x ?o)
  =>
        (retract ?fo ?fl)
        (assert (occ ?x (+ ?o 1)))
        (assert (lista $?s)))

(defrule min
        (declare (salience -1))
        (occ ?x ?m)
        (not (occ ? ?v & :(> ?v ?m)))
  =>
        (assert (min ?x ?m)))
```

7. Scrieti 3 patternuri CLIPS a.i. fiecare sa se potriveasca cu toate faptele urmatoare: (m 1 2 5 7 10 17), (m 1 5 9 17), (m 17 5 29). (4p+3p+3p)

(m $?)
(m $? 17 $?)
(m ? $? 5 $?)

8. Unificati urmatoarele 2 propozitii (5p), precizand substitutiile realizate (5p):
cunoscuti(prieten(george),Y,Z).
cunoscuti(X,frate(X),george).

S = {prieten(george)/X,frate(X)/Y,george/Z} =>
S = {prieten(george)/X,frate(prieten(george))/Y,george/Z} =>
cunoscuti(prieten(george),frate(prieten(george)),george)

9. Scrieti un program Prolog care realizeaza quicksort. (10p)

```
partition([], _, [], []).
partition([H|L], P, [H|L1], L2) :- H < P,  partition(L, P, L1, L2).
partition([H|L], P, L1, [H|L2]) :- H >= P, partition(L, P, L1, L2).

quickSort([], []).
quickSort([H|L], S) :-
        partition(L, H, L1, L2),
        quickSort(L1, S1),
        quickSort(L2, S2),
        append(S1, [H|S2], S).
```

10. Traduceti urmatoarea propozitie in FOL:

"Orice nas isi are nasul." (10p)

$\forall X.(\exists Y.nas(X,Y) => \exists Z.nas(Z,X))$