N3

**Programare Logica**

**%%Exercitiul 2**

%% Scrieti un predicat care recunoaste palindroame.

estePalindrom([]).

estePalindrom([H|T]):-

append(P,[H],T), estePalindrom(P).

**%%Exercitiul 3**

**%%** Sa se scrie un predicat pentru determinarea maximului unei liste de intregi

max([],A).

max([H|T],M):-

maxAcc(T,H,M).

maxAcc([],A,A).

maxAcc([H|T],A,M):-

H>A, maxAcc(T,H,M);

maxAcc(T,A,M).

**%%Exercitiul 4**

**%%** Sa se defineasca predicatul shift stg (List1, List2) astfel ca List2 este List1 ”cu un shift rotational” cu un element spre stanga

shift\_st([],[]).

shift\_st([H|T], L):-

append(T,[H],L).

**%%Exercitiul 5**

**%%** Sa se defineasca predicatul shift dr (List1, List2) astfel ca List2 este List1 ”cu un shift rotational” cu un element spre dreapta

shift\_dr([],[]).

shift\_dr(L, [H|T]):-

append(T,[H],L).

**%%Exercitiul 6**

**%%** Scrieti un program pentru calcului functiei factorial. Folositi acumulator

factorial(N, R) :- factorial(N, 1, R).

factorial(0, R, R) :- !.

factorial(N, Acc, R) :-

N1 is N - 1,

Acc1 is Acc \* N,

factorial(N1, Acc1, R).

**%%Exercitiul 9**

**%%** Scrieti un program suma si sumapatrate care dintr-o lista de numere calculeaza suma acestora si suma patratelor acestora. Folositi acumulatori.

suma\_si\_sumaPatrate([], 0, 0).

suma\_si\_sumaPatrate([H|T], S, Sp):-

suma\_si\_sumaPatrate(T, S1, Sp1), S is H+S1, Sp is Sp1+H\*H.

**%%Exercitiul 13**

**%%** Implementati algoritmul de sortare prin insertie liste de ıntregi in Prolog

sortareInserare([],[]).

sortareInserare([A],[A]).

sortareInserare([H|T], Out):-

sortareInserare(T,Z),

inserare(H,Z,Out).

inserare(X, [], [X]).

inserare(X, [H|T], [X,H|T]):-

X < H.

inserare(X, [H|T], [H|Z]):-

X >= H,

inserare(X,T,Z).

**%%Exercitiul 17**

**%%** Scrieti un predicat de 2 ori mai lung(L1,L2) care este satisfacut daca lista L2 este de doua ori mai lunga decat L1. Calcularea lungimii listelor nu este permisa.

de\_2\_ori\_mai\_lung([],[]).

de\_2\_ori\_mai\_lung([\_|L1], [\_,\_|L2]):-

de\_2\_ori\_mai\_lung(L1,L2).

**%%Exercitiul 18**

**%%** Scrieti predicatul fib (N,F) care este satisfacut daca F este al N-lea numar Fibonacci.

fib(1, 1) :- !.

fib(2, 1) :- !.

fib(N, F) :-

N > 2,

N1 is N-1,

N2 is N-2,

fib(N1, F1),

fib(N2, F2),

F is F1+F2.

**%%Exercitiul 23**

**%%** Scrieti un predicat ternar sterge primul(Element,Lista,Rezultat) care este satisfacut daca rezultatul este obtinut din lista prin stergerea primei aparitii a elementului.

sterge\_primul(\_, [], []).

sterge\_primul(X, [X|T], T).

sterge\_primul(X, [H|T], [H|R]):-

sterge\_primul(X,T,R).

**Programare Functionala**

**35.**Write a function even num(my−list) which returns the list of even numbers

from a list. Do not use the predicate even?. Exemple:

> ( even num ’ ( 1 2 3 4 4 5 7 8 ) )

( 2 4 4 8 )

Solution:

(define (elem-even ls)

(define res '()) ; definim lista de rezultat

(cond [(list? ls)

(for ([i ls]) ; iteram prin lista primita ca argument

(cond ((eq? (remainder i 2) 0) ;cand numarl e par il adaugam la lista de rezultat

(set! res (cons i res))

)))

(reverse res)]

[else "Argumentul nu este lista!"]))

(elem-even '(1 2 2 2 3 4 5 6))

(elem-even 67)

**12.** Write a tail recursive program which calculates the factorial of a natural number.

Solution:

(define (factorial n)

(if (number? n) ;Am verificat daca argumentul primit e numar

(if (= n 0) 1 ; cazul de baza

(\* n (factorial (- n 1)))

)

(quote "argumentul nu e numar"))

)

(factorial 6) ;720

**11.** Write a function shift right (List1) such that the returned list is List1 ”as

a rotational shift” with one element to the right. Example:

> ( shift\_right ’( 1 2 3 4 5 ) )

( 5 1 2 3 4 )

> ( shift\_right ’ ( 5 1 2 3 4 ) )

( 4 5 1 2 3 )

Solution:

(define (shift-right lst)

(if (list? lst) ;Am verificat daca argumentul primit e lista

(if (null? (cdr lst)) ; Am verificar daca lista e vida

lst

(cons (car (reverse lst)) (reverse (cdr (reverse lst)))))

(quote "argumentul nu e lista")))

(shift-right '(1 2 3 4 5))

**30.** Write a function delete all occ (Element,my-List) which deletes all the occurrences of an element from a list.

Exemplu:

> ( d e l e t e a l l o c c a ’ ( a b c a d a ) )

( b c d )

> ( d e l e t e a l l o c c a ’ ( b c d ) )

( b c d )

(define (delete-all-occ element my-list)

(cond

[(not (list? my-list)) (displayln "eroare")]

[(empty? my-list) empty] ; dacă lista este goală, returnăm lista goală

[(equal? (first my-list) element) (delete-all-occ element (rest my-list))] ;dacă primul element este egal cu elementul specificat, apelăm recursiv funcția pe restul listei

[else (cons (first my-list) (delete-all-occ element (rest my-list)))])) ; altfel, păstrăm primul element și apelăm recursiv funcția pe restul listei

(display (delete-all-occ 'a '(a b c a d a))) ;(b c d)

**15.** Write a program sum of which returns the sum of the elements from a list.

Write also the tail recursive version. Example:

> ( sum o f ’ ( 1 -3 2 0 ) )

0

(define (sum1 lst)

(cond

[(empty? lst) 0] ; daca lista este goală, suma este 0

[else (+ (first lst) (sum1 (rest lst)))])) ; Suma elementului curent cu suma celorlalte elemente

(display (sum1 '(1 -3 2 0)))

(define (sum2 lst acc)

(cond

[(empty? lst) acc] ; daca lista este goală, returnăm acumulatorul

[else (sum2 (rest lst) (+ acc (first lst)))])) ; Adăugăm elementul curent la acumulator și apelăm recursive

(display (sum2 '(1 2 3 4) 0)); 10

(display (sum1 '(1 -3 2 0))); 0

**14.** Write a program change string in order to change a string by transforming

all the vowels into V, all the consonants into C and all the remaining

characters into 0.

(define (change-v0c str)

(define (transform-char char)

(cond

[(regexp-match? #rx"[aeiouAEIOU]" (string char)) "V"] ; Dacă caracterul este o vocală, returnăm "V"

[(regexp-match? #rx"[a-zA-Z]" (string char)) "C"] ; Dacă caracterul este o consoană, returnăm "C"

[else "0"])) ; Altfel, returnăm "0"

; Convertim șirul într-o listă de caractere, aplicăm funcția transform-char pe fiecare caracter

; și apoi concatenăm rezultatele într-un șir final

(string-join (map transform-char (string->list str)) ""))

(display (change-v0c "exemplu exemplu"))

;VCVCCCV0VCVCCCV

**36.** Write a function odd num(my-list) which returns the list of odd numbers

from a list. Do not use the predicate odd?. Exemple:

4

> ( odd num ’ ( 1 2 3 4 4 5 7 8 ) )

( 1 3 5 7 )

(define (odds l)

(cond

((not (list? l)) "given argument is not a list!")

((null? l) '())

((list? (car l)) (append (odds (car l)) (odds (cdr l))))

(else (if (and (integer? (car l)) (not (= (modulo (car l) 2) 0)))

(cons (car l) (odds (cdr l)))

(odds (cdr l))

)

)

)

)

**9**. Write a function in Racket that returns the maximum element from a list of integers.

(define (range l)

(cond

((not (list? l)) "given argument is not a list!")

(else

(define maximum (car l))

(for ([i l])

(cond

((and (number? i) (> i maximum)) (set! maximum i))

)

)

maximum

)

)

)

;(range '(0 7 8 2 3 -1))

;(range 'a)

**13.** Write a program delete vowels (String) in order to delete the vowels from a string.

(define (member? u lst)

(not (equal? (member u lst) #f)))

(define (not-vowel? c)

(not (member? c (list #\A #\E #\I #\O #\U #\a #\e #\i #\o #\u))))

(define (remove-vowels s)

(list->string (filter not-vowel? (string->list s))))