Crăciun Ioan-Flavius
922
48
Gr

A. (DEFUN F2 (X, N)
(COND
((= N 1) 1)
(> X 2) (- N 2)
(> X 1) (X)
(T (- X 1))
)
)
(DEFUN F (N)
(F2 (F (- N 1)) N)
)
% ~~DEFUN F2 (X N)~~
% F(N) = F2(F(N-1), N)
% F2(X, N) = {
% 1, N = 1
% N-2, X > 2 and N ≠ 1
% X, X > 1 and X <= 2 and N ≠ 1
% ~~X~~
% X-1, else
%}

Răcim Ioan-Flavius

922

48

fr

B.

~~split([H|T],~~

~~split(T,~~

split([],L1,L2):-
    length(L1,0,N1),
    length(L2,0,N2),
    N1 <= N2,
    !.

split([H|T],[HL1],L2):-
    split(T,L1,L2).

split([H|T],L1,[H|L2]):-
    split(T,L1,L2).

%o split(L,L,L)
%o split(i,o,o)

length([],C,C).
length([_|T],C,R):-
    C1 is C+1,
    length(T,C1,R).

[L1|[L2]]

main(L,R):-
    split(L,L1,L2),
    check(L1,E),
    check(L2,E),

Păcuraru Ioan-Flavius
922
48
02 continuation B

~~check(L, Elem)~~

```
check([H, T], []):-
    check(T, H).

check([H|T], E):-
    check2(H, E),
    check(T, H).

check([], _).

gcd(0, X, X):- X > 0, !.
gcd(X, Y, Z):- X >= Y,
               X1 is X - Y,
               gcd(X1, Y, Z).

gcd(X, Y, Z):- X < Y,
               Y1 is Y - X,
               gcd(X1, X, Z).

check2(A, B):- gcd(A, B, 1).
```

My solution used the split predicate to get 2 partitions. Then I check the two partitions with a check predicate to check that the list's ~~elements~~ elements are relatively prime.

I know the check function is wrong because it only checks consecutive numbers but I had no time to write another predicate. Page 3

Crăciun Ioan-Flaviu
922
48
Cr

C. %enumerate(int, int)

%enumerate(i, o)

% this predicate returns successively n, n-1 down to 1

```
enumerate(1,1):-!.
enumerate(I,I).
enumerate(I,R):-
        I1 is I-1,
        enumerate(I1,R).

is_in([H|_],H):-!.
is_in([_|T],E):-
    is_in(T,E).
```

% this predicate returns true or false whether the integer is inside the list

% is_in(list, int)

% is_in(i, i)

```
ith_elem([H|_],I1,I1,H):-!
ith_elem([_|T],I1,I2,R):-
        I is I1+1,
        ith_elem(T,I,I2,R).
```

% this predicate returns the integer at position I inside the list akin to l[I]

% ith_elem(list, int, int, int)

% ith_elem(i, i, i, o)

CRĂCIUN IOAN-FLAVIU
922
48
CR

continuation C

```
% append(int, list, list)
% append(i, i, o)
append(H, T, [H, T]).

% length(list, int, int)
% length(i, i, o)
length([], C, C).
length([_|T], C, R):-
    C1 is C+1,
    length(T, C1, R).
% build_list(int, int, list)
% build_list(i, i, o)
% this predicate builds the list N, N+1 ... 2N-1 in reverse order
build_list(I, N, [E]):-
    I is 0,
    !,
    E is N.
build_list(I, N, L):-
    I1 is I-1,
    x is I+N,
    build_list(I1, N, L2),
    append(x, L2, L).
```

Crăciun Ioan-Flavius
922
48
continuation c

```
% permutations_aux(list,list,int,int,list,list,list or null)
% permutations_aux(i,i,i,i,i,0,i)
permutations_aux(_,_,0,_,Col,Col,_):-!.
permutations_aux(L,R,Rem,N,Col,Sol,Last):-
    enumerate(N,IDx),
    \+ is_in(R,IDx),
    ith_elem(L,1,IDx,Elem),
    check(Last,Elem),
    append(IDx,R,R1),
    append(Elem,Col,Col1),
    Rem1 is Rem-1,
    permutations_aux(L,R1,Rem1,N,Col1,Sol,Elem).
% check(int or null,int)
% check(i,i)
% this function returns ~~true~~ T/F is different <=> 2 or first int is []
check([],_):-!.
check(A,B):-
    A >= B-2,
    B >= A-2.
```

Bracim Ioan-Flavius
922
48
Pr
continuation c

```
main(N):-
    N1 is N-1,
    build-list(N1, N, L),
    permutations-aux(L, [], N, N, [], Sol, []).

%permutations-aux(L, R, Rem, N, Last) = {
        Sol, Rem=0
[iter-Rem(L, E, enumerate(N/as E) as EL] (+) permutations-aux(L, R+1E,
                                    Rem-1, N, EL, is-in(R, E)/EL
                                    check(Last, EL)
```

Crăciun Ioan-Flavius
922
48
6/2

D. (DEFUN myreplace (l level k)
(cond
~~((atom l))~~
((and (oddp level) (numberp l) (> l k)) (- l 1))
((atom l) l)
(t (mapcar #'(lambda (x) (myreplace x
(+ level 1) k)) l))
)
)
~~myreplace~~

(DEFUN main (l k) (myreplace l 0 k))

% myreplace(l, level, k) = {
l - 1, level %2 == 1 and l is number and l > k
l, atom and not number
$\bigcup_{i=1}^{n}$ myreplace(l_i, level+1, k), l is $l_1 l_2 \dots l_n$
}
% main(l k) = myreplace(l, 0, k)