Dan Cantor
922
DCant
Subject A

```
f([], -1).

f([H|T], S) :- f(T, S1), f_aux(H, S1, S).
```

```
% f_aux(H: integer, S: integer, R: integer)
% (i, i, o) - determinist
% f_aux is an auxiliary predicate which
% check the conditions for predicate f.
f_aux(H, S, H) :- H>0, S<H, !.
f_aux(H, S, S).
% By using this auxiliary predicate, we eliminate
% one of the calls of f(T, S). We will use the value computed
% for both branches of f_aux.
```

Dan Cantor
922
Dart
Subject B

$$insert(l_1 l_2 \ldots l_n, E, last) = \begin{cases} E \oplus l_1 l_2 \ldots l_n & \text{if } |E - l_1| \geq 2 \text{ and } |E - last| \geq 2 \\ l_1 \oplus insert(l_2 l_3 \ldots l_n, E, l_1) \end{cases}$$

% insert(L: list, E: integer, Last: integer, R: list)
% (i, i, i, o) - non-determinist
% insert value E on each position of list L if the absolute value
% between it and its neighbours is greater or equal to 2

```
insert([], E, Last, [E]) :-
    abs(E-Last) >= 2.
insert([H|T], E, Last, [E,H|T]) :-
    abs(E-Last) >= 2,
    abs(E-H) >= 2.
insert([H|T], E, _, [H|R]) :-
    Last1 is H,
    insert(T, E, Last1, R).
```

2/6

Dan Cantor
922
(2) A.apt
Subject B

$$\text{cond}(l_1 l_2 \ldots l_n) = \begin{cases} \text{return true if } n=0 \\ \text{return false if } |l_1 - l_2| < 2 \\ \text{return cond}(l_2 l_3 \ldots l_n) \text{ otherwise} \end{cases}$$

```
% cond (L : list)
% (i) - determinist
% check if a permutation satisfies the condition of the problem
cond([]).

cond([H1,H2|T]):-
    abs(H1-H2) >= 2,
    cond([H2|T]).
```

Dan Cantor
922

Plant

Subject B

$$perm(l_1 l_2 \ldots l_n) = \begin{cases} [\,], & \text{if } n=0 \\ insert(l_1, perm(l_2 \ldots l_n)), & \text{otherwise} \end{cases}$$

```
% perm(L: list, R: list)
% (i, o) - non-determinist
% Compute permutations of list L.

perm([], []).
perm([H|T], R):-
    perm(T, R1)
    insert(R1, H, H-2, R). % we put H-2 as last to make sure
                           % inserts pass the first test.
```

$$buildList(x) = \begin{cases} \text{if } x=0, \text{ return } [\,] \\ \text{return } buildList(x-1) \oplus X \end{cases}$$

```
% buildList(X: integer, R: list)
% (i, o) - determinist
% builds set 1..X
buildList(0, []).

buildsList(X, R):- X1 is X-1, buildsList(X1,R), R1#R,
    ~~buildsList(X-1, R~~ insertAtEnd(R1, X, R).
```

$$insertAtEnd(l_1 l_2 \ldots l_n, x) = \begin{cases} x \text{ if } n=0 \\ l_1 \oplus insertAtEnd(l_2 \ldots l_n, x) \end{cases}$$

Oan Cantor
922
Ocant
Subject B

% insertAtEnd(L: list, x: ~~number~~ integer, R: list)
% (i, i, o) - determinist
% inserts value at the end of a list.

```
insertAtEnd([ ],X,[X]).
insertAtEnd([H|T],X,[H|R]):-
    insertAtEnd(T,X,R).
```

$$oneSol(l_1 l_2 \ldots l_n) = \{ perm(l_1 l_2 \ldots l_n) \text{ if } cond(l_1 l_2 \ldots l_n)$$

% ~~oneSol(l1 l2 .. ln):~~ oneSol(L: list, R: list)
% (i, o) - non-determinist
% Get a solution of the problem.

```
oneSol(L, R):-
    perm(L, R),
    cond(R).
```

$$allSol(\cancel{l_1 l_2 \ldots l_n} n) = \bigcup oneSol(buildList(n)).$$

% allSol(X: ~~list~~ integer, R: list):
% (i, o) - determinist

```
allSol(X, R):- buildList(X, R1),
    findall(R2, oneSol(R1, R2), R).
```

Dan Cantor
922
DCant
Subject C

$$\text{replaceAtoms}(L, level) = \begin{cases} 0 \text{ if atom}(L) \text{ and } level \% 2 = 0, \\ L \text{ if atom}(L), \\ \bigcup_{i=1}^{n} \text{replaceAtoms}(L_i, level+1), \\ \quad \text{where } L_i \text{ is element of } L. \end{cases}$$

```
(DEFUN replaceAtoms (L level)
    (COND
        ((AND (ATOM L) (= (MOD level 2) 0)) 0)
        ((ATOM L) L)
        (T (MAPCAR #'(LAMBDA (X)
                          (replaceAtoms X (+ level 1))
                     )L
           )
        )
    )
)
(DEFUN wrapper (L)
  (replaceAtoms L 0)
)
```