Решения на задачите от контролно 2 по Логическо програмиране

18 май 2019

1 Първа задача на пролог

Да се дефинира на пролог предикат $q(N, [Ax, Ay], [Bx, By], [Cx, Cy], [Dx, Dy])$, който по дадено естествено число N генерира координатите на върховете на всички правоъгълници $ABCD$ в равнината ($AB \| DC \| \overrightarrow{\text{Ox}}$ и $AD \| BC \| \overrightarrow{\text{Oy}}$), чиито координати са естествени числа и:

I.1 периметъра им е N.

I.2 лицето им е N.

1.1 Общи предикати

```
natural(0).
natural(N) :-
    natural(M),
    N is M+1.

pairs(A, B) :-
    natural(N),
    between(0, N, A),
    B is N-A.
```

1.2 Примерно решение на I.1

```
generateSidesArea(N, A, B) :-
    between(1,N,A),
    0=:=N mod A,
    B is N div A,
    B>0.
q(N, [[Ax, Ay], [Bx, By], [Cx, Cy], [Dx, Dy]]) :-
    pairs(Ax,Ay),
    generateSidesPerimeter(N,SA,SB),
    Bx is Ax+SA,
    By is Ay,
    Cx is Ax+SA,
    Cy is Ay+SB,
    Dx is Ax,
    Dy is Ay+SB.
```

1.3 Примерно решение на I.2

```
generateSidesPerimeter(N, A, B) :-
    between(1,N,A1),
    A1 mod 2=:=0,
    A is A1 div 2,
    B1 is N-A1,
```

```
    0=:=B1 mod 2,
    B is B1 div 2,
    B > 0.

q(N, [[Ax, Ay], [Bx, By], [Cx, Cy], [Dx, Dy]]) :-
    pairs(Ax, Ay),
    generateSidesArea(N, SA, SB),
    Bx is Ax+SA,
    By is Ay,
    Cx is Ax+SA,
    Cy is Ay+SB,
    Dx is Ax,
    Dy is Ay+SB.
```

2 Втора задача на пролог

Да се на пролог дефинира предикат $p(X, Y)$, който по даден списък от естествени числа X намира списък от списъци Y , който съдържа всички пермутации на X и всеки елемент на Y се среща в Y точно толкова пъти, колкото е II.1 неговият последен елемент. II.2 сумата от елементите му.

2.1 Общи предикати

```
toSet([], []).
toSet([H|T], [H|R]) :-
    toSet(T, R),
    not(member(H, R)).
toSet([H|T], R) :-
    toSet(T, R),
    member(H, R).

select(X, L, R) :-
    append(A, [X|B], L),
    append(A, B, R).

insert(X, L, R) :-
    append(A, B, L),
    append(A, [X|B], R).

permutate([], []).
permutate(L, [H|R]) :-
    select(H, L, RL),
    permutate(RL, R).

packPermutations([], [], []).
packPermutations(L, R, R) :-
    not(( permutate(L, P),
            not(member(P, R))
        )).
packPermutations(L, R, S) :-
    permutate(L, P),
    not(member(P, R)),
    packPermutations(L, [P|R], S).

duplicate([], []).
duplicate([[0, _]|T], R) :-
    duplicate(T, R).
duplicate([[N, L]|T], R) :-
    N>0,
```

```
    N1 is N-1,
    duplicate([[N1, L]|T], R1),
    insert(L, R1, R).
```

2.2 Примерно решение на I.1

```
prepareForDuplication([], []).
prepareForDuplication([H|T], [[Last, H]|R]) :-
    prepareForDuplication(T, R),
    append(_, [Last], H).

p(L,R):-
    packPermutations(L, [], P),
    toSet(P, SP),
    prepareForDuplicationA(SP, PP),
    duplicate(PP, R).
```

2.3 Примерно решение на I.2

```
prepareForDuplication(L, R):-
    sum(L, S),
    prepareForDuplication(L, S, R).

prepareForDuplication([], _, []).
prepareForDuplication([H|T], S, [[S, H]|R]) :-
    prepareForDuplication(T, S, R).

sum([], 0).
sum([H|T], N):-
    sum(T, M),
    N is M + H.

p(L,R):-
    packPermutations(L, [], P),
    toSet(P, SP),
    prepareForDuplicationB(SP, PP),
    duplicate(PP, R).
```