Laborator 3

Logică matematică și computațională

Laboratorul 3

Cuprins

- Alte exerciții cu liste
- Sortări
- Arbori binari

Alte exerciții cu liste

Exercițiul 1
Definiți un predicat `palindome/1` care este adevărat dacă lista primită ca argument este palindrom (lista citită de la stânga la dreapta este identică cu lista citită de la dreapta la stânga).

De exemplu, la întrebarea

```
?-palindrome([r,e,d,i,v,i,d,e,r]).
```

ar trebui să obțineți `true`.

Nu folosiți predicatul predefinit `reverse`, ci propria implementare a acestui predicat.

Practică

Exercițiul 2
Definiți un predicat `remove_duplicates/2` care șterge toate duplicatele din lista dată ca prim argument și întoarce rezultatul în al doilea argument.

De exemplu, la întrebarea

```
?- remove_duplicates([a, b, a, c, d, d], List).
```

ar trebui să obțineți `List = [b, a, c, d]`.

Exercițiul 3
Definiți un predicat `atimes/3` care să fie adevărat exact atunci când elementul din primul argument apare în lista din al doilea argument de numărul de ori precizat în al treilea argument.

Interogați:

```
?- atimes(3,[3,1,2,1],X).
?- atimes(1,[3,1,2,1],X).
?- atimes(N,[3,1,2,1],2).
?- atimes(N,[3,1,2,1],1).
?- atimes(N,[3,1,2,1],0).
?- atimes(N,[3,1,2,1],X).
```

Sortări

Sortarea prin inserție (*insertion sort*)

Predicatul `insertsort/2` sortează lista de pe primul argument folosing algoritmul *insertion sort*.

```
insertsort([],[]).
insertsort([H|T],L) :- insertsort(T,L1), insert(H,L1,L).
```

Exercițiul 4: scrieți regulile care definesc comportamentul predicatului ajutător `insert/3`.

Quicksort

Predicatul `quicksort/2` sortează lista de pe primul argument folosing algoritmul *quicksort*.

```
quicksort([],[]).
quicksort([H|T],L) :-
split(H,T,A,B), quicksort(A,M), quicksort(B,N),
append(M,[H|N],L).
```

Exercițiul 5: scrieți regulile care definesc comportamentul predicatului ajutător `split/4`.

Arbori binari

Arbori binari – intuitiv

Reamintim că, în Prolog, listele erau de două feluri:

- lista vidă `[]`;
- listă nevidă `[H|T]` formată din capul `H` (care putea fi orice) și coada `T` (care era o listă).

Similar, arborii binari vor fi de două feluri:

- arborele vid;
- arbore nevid format dintr-o rădăcină și doi subarbori.

Arborii binari nu sunt o funcționalitate inerentă a Prolog-ului, de aceea pentru ei se vor folosi simboluri de funcție *ad hoc*, lucru permis de limbaj.

Arbori binari – definire și parcurgere

Pentru arborele vid, vom folosi atomul `vid`. Pentru arborele care are rădăcina `R` și cei doi subarbori `T` și `U`, vom folosi termenul `arb(R,T,U)`.

Interogați:

```
X = arb(3,arb(2,vid,vid),vid).
```

Definirea predicatului `srd/2` folosit la parcurgerea unui arbore binar în inordine:

```
srd(vid,[]).
srd(arb(R,T,U),L) :- srd(T,L1), srd(U,L2),
append(L1,[R|L2],L).
```

Testați acest predicat!

Exercițiul 6: arbori binari de căutare
Definiți următoarele predicate:

- `bt_ins/3` care inserează numărul natural din primul argument în arborele binar de căutare din al doilea argument;
- `bt_list/2` care transformă lista din primul argument într-un arbore binar de căutare;
- `bt_sort/2` care sortează lista din primul argument trecând prin arborele binar de căutare asociat ei.