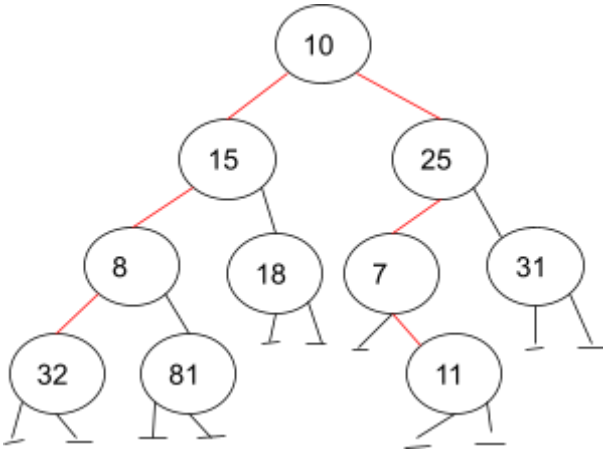
Nota 5. Dându-se un număr natural, sa se scrie un predicat *unique_digits* care colectează într-o lista cifrele numărului, fără duplicate.
?-unique_digits(71107, R)
R=[0,1,7]. (sau orice permutare)

Nota 6-7. Dându-se o lista adanca, să se scrie un predicat care aplatizeaza lista un nivel de la exterior spre interior și calculeaza produsul atomilor rezultați.
flatten_1([1,[2,3,[4,5], 6, [7], [8,[9,10,[11], 12]], 3], [[14]], R, P).
R = [1, 2, 3, [4,5], 6, [7], [8, [9,10,[11],12]], 3, [14]], P = 108.

Nota 7-8. Dându-se o lista adanca, să se extragă acele liste si subliste care conțin o singura lista și în rest atomi.
?-one_deep([0,[1,2,[1,2,[1,2,[1,2], 3], 4, [5,6,[7]]], 8],1] R).
R=[
[1,2,[1,2,[1,2,[1,2], 3], 4, [5,6,[7]]], 8],
[1,2,[1,2], 3],
[5,6,[7]]
]

Nota 9. Dându-se un arbore binar terminat în variabila liberă, scrieți un predicat care colectează în ordine nodurile de pe cel mai lung lanț.
tree(t(10, t(15, t(8, t(32, _, _), t(81, _, _)), t(18, _, _)), t(25, t(7, _, t(11, _, _)), t(31, _,_)))).
?-tree(T), longest_chain(T, R).
R = [32,8,15,10,25,7,11]. (dacă sunt mai multe soluții se accepta oricare)



Nota 10. Avand o baza de cunoștințe care la apelul *anumber(X)*, unifica X pe rand cu cate un număr natural, scrieți un predicat *group_numbers* (și orice alte predicate ajutătoare) care modifica baza de cunoștințe astfel incat sa pastreze cunoștințele existente, la care adaugă fapte de forma numbers(A,K), astfel incat la apelul *numbers(A,K)*, A să se unifice cu lista numerelor de K cifre (ordinea în A fiind ordinea inițială din baza de cunoștințe, fără duplicate)
Inițial
number(10). number(198). number(45). number(2000), number(1). number(62). number(9). number(540). number(861). number(111). number(10).

?-group_numbers, numbers(A,3), numbers(B, 2).
A = [198, 540, 861, 111], B = [10, 45, 62].