Tema 3

1. Exploraţi [1] teoria numerelor naturale folosind Prolog. Spre aducere aminte,

   - 0 este un număr natural,
   - dacă x este un număr natural, atunci si $s(x)$ (succesorul său) este, şi
   - acestea se definesc în Prolog ca un predicat unar `nat/1` care este adevărat când argumentul său este un număr natural:

     ```
     nat(0).
     nat(s(X)):-nat(X).
     ```

   Cu 0 şi s, funcţia succesor, se poate defini suma $+$ a două numere naturale x, y:

   $$0 + y = y,$$
   $$s(x) + y = s(x + y).$$

   În Prolog (unde există doar predicate), se introduce predicatul ternar `pluss/3`, care spune ca suma primelor două argumente este al treilea:

   ```
   pluss(0, Y, Y) :-
                   nat(Y).
   pluss(s(X), Y, s(Z)) :-
                 pluss(X, Y, Z).
   ```

2. Scrieţi un predicat care recunoaşte palindroame[2].

3. Să se scrie un predicat pentru determinarea maximului unei liste de întregi.

4. Să se definească predicatul `shift_stg(List1, List2)` astfel ca List2 este List1 "cu un shift rotaţional" cu un element spre stânga. Exemplu:

   ```
   ?- shift_stg([1,2,3,4,5], L1), shift_stg(L1,L2).

   L1=[2,3,4,5,1]
   L2=[3,4,5,1,2];

   No
   ```

5. Să se definească predicatul `shift_dr(List1, List2)` astfel ca List2 este List1 "cu un shift rotaţional" cu un element spre dreapta. Exemplu:

   ```
   ?- shift_dr([1,2,3,4,5], L1), shift_dr(L1,L2).

   L1=[5,1,2,3,4]
   L2=[4,5,1,2,3];

   No
   ```

[1] Definiţi adunarea (deja făcută ca exemplu), înmulţirea, exponenţierea, mai mic, mai mic egal, divide, minus (atentie, numai varianta binara are sens), diviziunea

[2] Un palinfrom este un cuvânt, frază, număr (sau orice altă secvenţă de obiecte) care are proprietatea că citit/ă (parcurs/ă) din orice direcţie arată la fel (ajustarea punctuaţiei şi spaţiilor dintre cuvinte este permisă). [sursa:wikipedia.org]

6. Scrieţi un program pentru calcului funcţiei factorial. Folosiţi acumulator.

7. Scrieţi un program sterge_vocale(Sir, SirFaraVocale) care şterge vocalele dintr-un şir de caractere.

8. Scrieţi un program schimba_sir care schimbă un şir de caractere prin transformarea tuturor vocalelor în reprezentarea lor cu literă mare, toate consoanele în reprezentarea lor cu literă mică şi toate celelalte caractere în 0.

9. Scrieţi un program suma_si_sumapatrate care dintr-o listă de numere calculeaza suma acestora şi suma pătratelor acestora. Folosiţi acumulatori. Exemplu Use accumulators. Example:

```
?- suma_si_sumapatrate([1,-3,2,0], Sum, SQS).

Sum = 0
SQS = 14 ;

No
```

10. Definiţi o relaţie binară prefix/2 între liste şi prefixele acestora. Sugestie: [], [a] şi [a, b] sunt prefixele listei [a, b].

11. Definiţi o relaţie binară sufix/2 între liste şi toate sufixele lor. Sugestie: [], [b] şi [a, b] sunt sufixele listei [a, b].

12. Definiţi o relaţie binară sublista/2 între liste si sublistele lor.

13. Implementaţi algoritmul de sortare prin inserţie liste de întregiîn Prolog – informal, acesta poate fi formulat astfel:

    Fiind dată o listă, se elimină capul ei, se sortează coada, iar apoi capul se introduce în lista sortată într-o poziţie ce păstrează lista sortată.

14. Implementaţi algoritmul de sortare prin selecţie pentru liste de întregi în Prolog – informal, acesta poate fi formulat astfel:

    Fiind dată o listă, se găseşte elementul minim, se plasează pe prima poziţie şi se repetă procesul pentru coada listei.

15. Implementaţi algoritmul de sortare rapidă pentru o listă de întregi în Prolog – informal, acesta poate fi formulat astfel:

    Fiind dată o listă, se împarte în două – o parte conţine elemente mai mici decât un element al listei (pivot) iar cealaltă conţine elementele mai mari. Cele două părţi se sortează iar variantele sortate se concatenează.

16. Implementaţi algoritmul de sortare prin intercalare pentru o listă de întregi în Prolog – informal, acesta poate fi formulat astfel:

    Fiind dată o listă, de împarte în două părţi de mărimi egale. Se sortează cele două părţi, iar rezultatele sortate se combină prin intercalare astfel ca ordinea elementelor să se păstreze.

17. Scrieţi un predicat de_2_ori_mai_lung(L1,L2) care este satisfăcut dacă lista L2 este de două ori mai lungă decât L1. Calcularea lungimii listelor nu este permisă.

18. Scrieţi predicatul fib(N,F) care este satisfă cut dacă F este al N-lea număr Fibonacci[3]. Să se calculeze fib(5,F), fib(10,F), fib(50,F).

19. Implementaţi algoritmul euclidian extins[4] pentru calculul celui mai mare divizor comun a două numere întregi.

20. Scrieţi un predicat fara_dubluri_1(Xs, Ys) care este adevărat dacă Ys este lista de elemente din Xs fără dubluri. Elementele să fie în aceeaşi ordine în Ys şi Xs, păstrându-se ultima apariţie a elementelor duplicat.

    Exemplu:

```
?- fara_dubluri_1([1,2,3,4,5,6,4,4],X).
   X = [1, 2, 3, 5, 6, 4];

   No
```

21. Scrieţi un predicat fara_dubluri_2(Xs, Ys) care este adevărat dacă Ys este lista de elemente din Xs fără dubluri. Elementele să fie în ordine inversă în Ys faţă de apariţia lor înXs, păstrându-se prima apariţie a elementelor duplicat.

    Exemplu:

```
?- fara_dubluri_2([1,2,3,4,5,6,4,4],X).
   X = [6, 5, 4, 3, 2, 1];

   No
```

22. Scrieţi un predicat ternar sterge_tot(Element,Lista,Rezultat) care este satisfăcut dacă rezultatul este obtinut din listă prin ştergerea tuturor aparitiilor elementului.

    Exemplu:

```
?-sterge_tot(a,[a,b,c,a,d,a],X).
  X=[b,c,d]


?-sterge_tot(a,[b,c,d],X).
  X=[b,c,d]
```

23. Scrieţi un predicat ternar sterge_primul(Element,Lista,Rezultat) care este satisfăcut dacă rezultatul este obtinut din listă prin ştergerea primei aparitii a elementului.

```
?-sterge_primul(a,[a,b,c,a,d,a],X).

X=[b,c,a,d,a]


?-sterge_primul(a,[b,c,d],X).

X=[b,c,d]
```

[3] Vezi http://en.wikipedia.org/wiki/Fibonacci_number
[4] Vezi http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm

24. Scrieţi un predicat binar numara_aparitii(Input,Rezultat) care este adevărat dacă Rezultat este o listă perechi [element, numar_de_aparitii_in_Input], unde 'element' este un element din lista Inputis iar numar_de_aparitii_in_Input este un întreg ce dă numărul de apariţii ale elementului. O astfel de pereche trebuie să apară în rezultat pentru fiecare element al listei Input.

    Exemplu:

```
?- numara_aparitii([a,b,a,a,b,c],X).
  X = [[c, 1], [b, 2], [a, 3]] ;

No
```

25. Scrieţi un predicat ternar sterge_nlea care şterge fiecare al N-lea element dintr-o listă.

    Exemple:

```
?- sterge_nlea([a,b,c,d,e,f],2,L).

L = [a, c, e] ;

No
?- sterge_nlea([a,b,c,d,e,f],1,L).

L = [] ;

No
?- sterge_nlea([a,b,c,d,e,f],0,L).

No
?- sterge_nlea([a,b,c,d,e,f],10,L).

L = [a, b, c, d, e, f] ;

No
```