

Tema suplimentară 1

1. Scrieți o funcție în Racket care să simuleze comportamentul lui CAR (fără a folosi CAR).
2. Scrieți o funcție în Racket care să simuleze comportamentul lui CDR (fără să folosiți CDR).
3. Scrieți o funcție în Racket care să simuleze comportamentul lui CONS (fără să folosiți CONS).
4. Scrieți o funcție în Racket care să simuleze comportamentul lui SECOND (fără să folosiți SECOND).
5. Scrieți o funcție în Racket care să simuleze comportamentul lui THIRD (fără să folosiți THIRD).
6. Scrieți o funcție în Racket care să simuleze comportamentul lui + (fără să folosiți +).
7. Explorați ¹ teoria numerelor naturale folosind Racket. Spre aducere aminte,
 - 0 este un număr natural,
 - dacă x este un număr natural, atunci și $s(x)$ (succesorul său) este, și
 - acestea se definesc în Lisp ca o funcție `nat/1` care este adevărată când argumentul său este un număr natural:

```
0 e numar natural
Daca x e natural , atunci si succesorul lui x
e numar natural.
```

Cu 0 și s , funcția succesor, se poate defini suma + a două numere naturale x, y :

$$\begin{aligned}0 + y &= y, \\ s(x) + y &= s(x + y).\end{aligned}$$

8. Scrieți o funcție care recunoaște palindroame².
9. Să se scrie o funcție pentru determinarea maximului unei liste de întregi.
10. Să se definească o funcție `shift_stg (List1)` astfel ca lista returnată este List1 "cu un shift rotațional" cu un element spre stânga. Exemplu:

```
> (shift_stg '(1 2 3 4 5))
(2 3 4 5 1)

> (shift_stg '(2 3 4 5 1))
(3 4 5 1 2)
```

¹Definiți adunarea (deja făcută ca exemplu), înmulțirea, exponențierea, mai mic, mai mic egal, divide, minus (atenție, numai varianta binară are sens), diviziunea

²Un palindrom este un cuvânt, frază, număr (sau orice altă secvență de obiecte) care are proprietatea că citit/ă (parcurs/ă) din orice direcție arată la fel (ajustarea punctuației și spațiilor dintre cuvinte este permisă). [sursa:wikipedia.org]

11. Să se definească o funcție `shift_dr (List1)` astfel ca lista returnată este List1 "cu un shift rotațional" cu un element spre dreapta. Exemplu:

```
> (shift_dr '(1 2 3 4 5))  
(5 1 2 3 4)
```

```
> (shift_dr '(5 1 2 3 4))  
(4 5 1 2 3)
```

12. Scrieți un program pentru calculul funcției factorial. Să fie eficient (final recursiv).
13. Scrieți un program `sterge_vocale (Sir)` care șterge vocalele dintr-un șir de caractere.
14. Scrieți un program `schimba_sir` care schimbă un șir de caractere prin transformarea tuturor vocalelor în reprezentarea lor cu literă mare, toate consoanele în reprezentarea lor cu literă mică și toate celelalte caractere în 0.
15. Scrieți un program `suma` care dintr-o listă de numere calculează suma acestora. Scrieți și varianta final recursivă. Exemplu:

```
> (suma '(1 -3 2 0))  
0
```

16. Scrieți un program `suma_patrate` care dintr-o listă de numere calculează suma pătratelor acestora. Scrieți și varianta final recursivă. Exemplu:

```
> (suma_patrate '(1 -3 2 0))  
14
```

17. Scrieți un program `media_aritmetica` care dintr-o listă de numere calculează media aritmetică a acestora. Scrieți și varianta final recursivă. Exemplu:

```
> (media_aritmetica '(1 -10 2 9))  
1
```

18. Definiți o relație binară `prefix/2` între liste și prefixele acestora. Sugestie: `()`, `(a)` și `(a b)` sunt prefixele listei `(a b)`.
19. Definiți o relație binară `sufix/2` între liste și toate sufixele lor. Sugestie: `()`, `(b)` și `(a b)` sunt sufixele listei `(a b)`.
20. Definiți o relație binară `sublista/2` între liste și sublistele lor.
21. Implementați algoritmul de sortare prin inserție liste de întregi în Racket – 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 poziție ce păstrează lista sortată.
22. Implementați algoritmul de sortare prin selecție pentru liste de întregi în Racket – informal, acesta poate fi formulat astfel:
Fiind dată o listă, se găsește elementul minim, se plasează pe prima poziție și se repetă procesul pentru coada listei.

23. Implementați algoritmul de sortare rapidă pentru o listă de întregi în Racket – informal, acesta poate fi formulat astfel:
Fiind dată o listă, se împarte în două – o parte conține elemente mai mici decât un element al listei (pivot) iar cealaltă conține elementele mai mari. Cele două părți se sortează iar variantele sortate se concatenează.
24. Implementați algoritmul de sortare prin intercalare pentru o listă de întregi în Racket – informal, acesta poate fi formulat astfel:
Fiind dată o listă, de împarte în două părți de mărimi egale. Se sortează cele două părți, iar rezultatele sortate se combină prin intercalare astfel ca ordinea elementelor să se păstreze.
25. Scrieți o funcție `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.
26. Scrieți o funcție `fib(N,F)` care este satisfăcută dacă F este al N-lea număr Fibonacci³. Să se calculeze `(fib 5)`, `(fib 10)`, `(fib 50)`. Varianta final recursivă.
27. Implementați algoritmul euclidian extins⁴ pentru calculul celui mai mare divizor comun a două numere întregi.
28. Scrieți o funcție `fara_dubluri_1 (lista)` care este returnează lista fără dubluri. Elementele să fie în aceeași ordine în lista returnată ca și în lista inițială, păstrându-se ultima apariție a elementelor duplicat.
Exemplu:

```
> (fara_dubluri_1 '(1 2 3 4 5 6 4 4))
(1 2 3 5 6 4)
```
29. Scrieți o funcție `fara_dubluri_1 (lista)` care este returnează lista fără dubluri. Elementele să fie în ordine inversă în lista returnată ca și în lista inițială, păstrându-se prima apariție a elementelor duplicat.
Exemplu:

```
> (fara_dubluri_2 '(1 2 3 4 5 6 4 4))
(6 5 4 3 2 1)
```
30. Scrieți o funcție `sterge_tot (Element,Lista)` care șterge toate aparițiile lui Element în listă.
Exemplu:

```
> (sterge_tot a '(a b c a d a))
(b c d)

> (sterge_tot a '(b c d))
(b c d)
```

³Vezi http://en.wikipedia.org/wiki/Fibonacci_number

⁴Vezi http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm

31. Scrieți o funcție `sterge_primul(Element,Lista)` care șterge prima apariție a elementului din listă.

```
> (sterge_primul a '(a b c a d a))  
(b c a d a)  
  
> (sterge_primul a '(b c d))  
(b c d)
```

32. Scrieți o funcție `numara_aparitii (Lista)` care returnează listă perechi [element, numar_de_aparitii_in_lista], unde 'element' este un element din lista Lista, iar `numar_de_aparitii_in_lista` este un întreg ce dă numărul de apariții ale elementului. O astfel de pereche trebuie să apară în rezultat pentru fiecare element al listei.

Exemplu:

```
> (numara_aparitii '(a b a a b c))  
((c 1) (b 2) (a 3))
```

33. Scrieți o funcție `pozitie_element (Lista)` care returnează poziția pe care se află un element într-o listă.

Exemplu:

```
> (pozitie_element 2 '(1 2 3 4 5))  
1  
  
>(pozitie_element 2 '(1 2 3 2 4 5))  
1
```

34. Scrieți o funcție `sterge_nlea (lista , n)` care șterge fiecare al N-lea element dintr-o listă.

Exemple:

```
> (sterge_nlea '(a b c d e f) 2)  
(a c e)  
  
> (sterge_nlea '(a b c d e f) 1)  
nil  
  
> (sterge_nlea '(a b c d e f) 0)  
nil  
  
> (sterge_nlea '(a b c d e f) 10)  
(a b c d e f)
```

35. Scrieți o funcție `pare(lista)` care returnează lista cu numerele pare dintr-o listă. Este interzisă folosirea predicatului `even?`. Exemple:

```
> (pare '(1 2 3 4 4 5 7 8))  
(2 4 4 8)
```

36. Scrieți o funcție `impare(lista)` care returnează lista cu numerele impare dintr-o listă. Este interzisă folosirea predicatului `odd?`. Exemple:

```
> (impare '(1 2 3 4 4 5 7 8))
(1 3 5 7)
```

37. Scrieți o funcție `modul(lista)` care returnează lista cu numerele în valoare absolută dintr-o listă. Este interzisă folosirea funcției `ABS`. Exemple:

```
> (modul '(1 -2 3 -4 4 5 -7 8))
(1 2 3 4 4 5 7 8)
```

38. Scrieți o funcție `lungimi-egale(lista1, lista2)` care returnează `true` dacă cele două liste au aceeași lungime și `false` altfel.

39. Scrieți o funcție `ordine(elem1, elem2, lista)` care returnează `true` dacă `elem1` se află înaintea lui `elem2` în lista și `false` altfel.

Exemple:

```
> (ordine 1 2 '(1 2 3 4 4 5 7 8))
T
```

```
> (ordine 'a 'b '(d a c f g glg b))
T
```

```
> (ordine 'a 'b '(o y b l a c f g glg))
NIL
```

```
> (ordine 'a 'b '(d a c f g glg))
NIL
```

40. Scrieți o funcție `elem_egale(lista1, lista2)` care returnează `T` dacă `lista1` și `lista2` au aceleași elemente.

Exemple:

```
> (elem_egale '(1 2 3 4) '(1 2 3 4))
T
```

```
> (elem_egale '(a d c) '(a d c))
T
```

```
> (elem_egale '(a c d) '(a c))
NIL
```

```
> (elem_egale '(a c) '(a c d))
NIL
```

```
> (elem_egale '() '(1))
NIL
```

```
> (elem_egale '() '())
NIL
```