

КОНТРОЛНА РАБОТА № 1 ПО ФУНКЦИОНАЛНО ПРОГРАМИРАНЕ
КН, 2-ри курс, 1-ви поток (28.11.2020 г.)

Задача 1. Някои числа имат интересни свойства. Например:

$$\begin{aligned}89 &\rightarrow 8^1 + 9^2 = 89 = 89 * 1 \\695 &\rightarrow 6^2 + 9^3 + 5^4 = 1390 = 695 * 2 \\46288 &\rightarrow 4^3 + 6^4 + 2^5 + 8^6 + 8^7 = 2360688 = 46288 * 51\end{aligned}$$

Да се дефинира процедура (`dig-pow n p`), която приема естествено число n (записано с цифри $abcd\dots$, които могат да се повтарят) и намира естествено число k - такова, че $(a^p + b^{p+1} + c^{p+2} + d^{p+3} + \dots) = n * k$. Ако число k с посоченото свойство не съществува, да се връща -1 .

Примери:

```
(dig-pow 89 1)      ; -> 1  (81 + 92 = 89 = 89 * 1)
(dig-pow 92 1)      ; -> -1 (няма k - такова, че 91 + 22 = 92 * k)
(dig-pow 695 2)     ; -> 2  (62 + 93 + 54 = 1390 = 695 * 2)
(dig-pow 46288 3)   ; -> 51 (43 + 64 + 25 + 86 + 87 = 2360688 =
                        46288 * 51)
```

Задача 2. Да се дефинира процедура (`kth-max-min xs`), която приема списък от цели числа и връща процедура с параметър естествено число k - такова, че оценката на израза $((kth-max-min xs) k)$ е k -тото по големина отрицателно число в xs . Ако такова не съществува, да се връща грешката "No such number".

Примери:

```
((kth-max-min '(1 2 3 4 -5 6 7 -2 -1 0)) 2) ; -> -2
((kth-max-min '(-1 0 -1 0 -2 3 1 -1)) 3)    ; -> No such number
```

Задача 3. Да се дефинира процедура (`shuffle xs`), която получава списък от $2*n$ елемента във вида $(x_1 x_2 \dots x_n y_1 y_2 \dots y_n)$ и връща списък във вида $(x_1 y_1 x_2 y_2 \dots x_n y_n)$.

Примери:

```
(shuffle '(2 5 1 3 4 7)) ; -> '(2 3 5 4 1 7)
(shuffle '(1 2 3 4 4 3 2 1)) ; -> '(1 4 2 3 3 2 4 1)
(shuffle '(1 1 2 2)) ; -> '(1 2 1 2)
```

Задача 4. Да се дефинира предикат `(triangular? mat)`, който получава квадратна числова матрица, представена като списък от списъци, и проверява дали тя е горно триъгълна, т.е. дали всичките елементи под главния ѝ диагонал са нули.

Примери:

```
(triangular? '((1 2 3)
               (0 5 6)
               (0 0 9))) ; -> #t
```

```
(triangular? '((0 2 3)
               (0 0 6)
               (1 0 0))) ; -> #f
```

```
(triangular? '((1 2 3)
               (1 5 6)
               (0 0 9))) ; -> #f
```

```
(triangular? '((1 2 3 4)
               (0 5 6 7)
               (0 0 8 9)
               (0 0 0 9))) ; -> #t
```