КОНТРОЛНА РАБОТА № 1 ПО ФУНКЦИОНАЛНО ПРОГРАМИРАНЕ
КН, втори курс, първи поток (18.11.2023 г.)

Задача 1

Дефинирайте процедура **(right-max xs)**, която получава списък от числа `xs` и връща нов списък със същата дължина, в който всяко от числата от `xs` е заменено с максималния елемент на подсписъка на `xs`, съставен от това число и всички числа, които са вдясно от него (числата от `xs` с индекс, по-голям или равен на текущия).

Примери:

```
(right-max '(1 2 3 4 -5 6 7 -2 -1 0)) ⟶ '(7 7 7 7 7 7 7 0 0 0)
(right-max '(5 8 9 12))               ⟶ '(12 12 12 12)
(right-max '(4 3 2 1 0))              ⟶ '(4 3 2 1 0)
```

Задача 2

Дефинирайте процедура **(kth-number xs)**, която приема списък от цели числа и връща двуаргументна процедура с параметри предикат `p` и естествено число `k` – такава, че оценката на израза `((kth-number xs) p k)` е `k`-тото по големина число `x` от списъка `xs`, за което е в сила `(p x)`. Ако такова число не съществува, да се породи грешка с текст "`No such number`".

Примери:

```
((kth-number '(1 2 3 4 -5 6))       odd? 2)      ⟶ 1
((kth-number '(1 2 3 4 -5 6))       negative? 1) ⟶ -5
((kth-number '(1 2 3 4 -5 -5 6))    negative? 2) ⟶ -5
((kth-number '(1 -4 2 3 4 -5 -5 6)) negative? 3) ⟶ -4
((kth-number '(-1 0 -1 0 -2)) negative? 4) ⟶ error "No such
number"
```

Задача 3

Дефинирайте процедура **(palindromize n)**, която получава естествено число `n`, състоящо се от четен брой цифри, от които може да се състави палиндром. Процедурата трябва да върне най-малкия палиндром, който може да се състави от цифрите на `n`.

Подсказка: Помислете как представянето на числото като сортиран списък от неговите цифри може да помогне за решаването на задачата.

Примери:

```
(palindromize 11)              ⟶ 11
```

```
(palindromize 3354457878)      ⟶ 3457887543
(palindromize 11335445789789) ⟶ 13457899875431
```

Задача 4

Разглеждаме числата от редицата на Фибоначи (**индексирана от 0**): 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

Вашата задача е:

(1) да генерирате `n`-тото число от редицата на Фибоначи, **посредством линеен итеративен процес**;
(2) да разбиете полученото число на *подгрупи* с дължина `k`. Възможно е подгрупата, образувана от най-десните цифри, да има дължина, по-малка от `k`. Възможно е и подгрупа да започва с 0;
(3) за всяка от подгрупите да намерите и върнете *максималната двойка*. Максималната двойка за дадена подгрупа е точкова двойка с най-често срещаната цифра и броя на срещанията ѝ в подгрупата. Ако няколко цифри се срещат най-често, да се върне най-малката.

Дефинирайте процедура от по-висок ред **`(around-fib n)`**, която приема неотрицателно цяло число `n` и връща едноаргументна процедура на естественото число `k`, която от своя страна връща списък от максималните двойки за всяка подгрупа с дължина `k` на числото на Фибоначи с индекс `n`. При конструирането на подгрупите с дължина `k` е необходимо да бъдат отчетени граничните случаи, посочени в (2).

Примери:

```
((around-fib 100) 25) ⟶ '((1 . 3))

((around-fib 180) 25) ⟶ '((1 . 5) (7 . 3))

((around-fib 1700) 25) ⟶ '((1 . 4) (2 . 5) (0 . 6) (4 . 5) (5 . 7)
(2 . 4) (6 . 7) (3 . 5) (0 . 4) (8 . 5) (4 . 5) (4 . 4) (7 . 7)
(7 . 6) (2 . 2))

((around-fib 500) 42) ⟶ '((0 . 6) (2 . 7) (2 . 6))

((around-fib 6000) 242) ⟶ '((5 . 31) (8 . 33) (8 . 31) (7 . 35)
(7 . 31) (4 . 7))
```