# Лабораторная работа № 5

## Задания на рекурсию

- 3. Напишите рекурсивную функцию recursion_max, на вход которой подается целочисленный список. Функция должна вернуть значение максимального элемента списка
- 6. Напишите рекурсивную функцию вычисления числа Фибоначчи fibonacci, на вход которой подается целочисленное значение n.
- 7. Напишите рекурсивную функцию pow_n, на вход которой подается два значения n и k (по умолчанию равно 8). Функция должна возвращать значение следующего вида: $n^k$.
- 15. Напишите рекурсивную функцию num_reverse, на вход которой подается целочисленное значение. Функция должна возвращать число, где цифры расположены в обратном порядке. Например, 1456 → 6541.

```
from typing import List, Optional


def recursion_max(ls: List[int], index: int = 0, mx: Optional[int] = None)
-> int:
    '''
    recursion method for search max in array
    '''
    if index == len(ls):
        return mx
    if mx is None:
        mx = ls[index]
    return recursion_max(ls, index+1, max(mx, ls[index]))

def fibonacci(count: int) -> int:
    '''
    recursion method for search fibocci number
    '''
    if count < 0:
        return 0
    if count <= 1:
        return 1
    return fibonacci(count-1) + fibonacci(count-2)

def pow_n(base: int, power: int = 8) -> int:
    '''
    raises number base to the power
    '''
```

```
    if power == 0:
        return 1
    if power == 1:
        return base
    if power % 2 == 0:
        return pow_n(base, power // 2) ** 2
    return pow_n(base, power-1) * base

def num_reverse(number: int, param: int = 0):
    '''
    reverse number: 1234 -> 4321
    '''
    if number == 0:
        return param
    last_cifr = number % 10
    return num_reverse(number // 10, param*10+last_cifr)
```

```
import unittest
import recursions as rec


class TestRecursions(unittest.TestCase):
    def test_recursion_max(self):
        test_cases = [
            ([1, 2, 3, 1, 2, 3], 3),
            ([], None),
            ([1, 1, 1], 1),
            ([10, 1, 2, 3], 10)
        ]
        for test in test_cases:
            test_case, right_answer = test
            mx = rec.recursion_max(test_case)
            self.assertEqual(mx, right_answer)

    def test_fibonacci(self):
        test_cases = [
            (-1, 0),
            (0, 1),
            (1, 1),
            (2, 2),
            (3, 3),
            (4, 5),
            (5, 8),
            (9, 55)
        ]
        for test in test_cases:
            test_case, right_answer = test
            self.assertEqual(rec.fibonacci(test_case), right_answer)
```

```
    def test_pow_n(self):
        test_cases = [
            ([1, 1], 1),
            ([2, 0], 1),
            ([3, 2], 9),
            ([2], 256),
            ([3, 3], 27)
        ]
        for test in test_cases:
            test_case, right_answer = test
            self.assertEqual(rec.pow_n(*test_case), right_answer)

    def num_reverse(self):
        test_cases = [
            (123, 321),
            (111, 111),
            (4224, 4224),
            (4, 4),
            (123456789, 987654321)
        ]
        for test in test_cases:
            test_case, right_answer = test
            self.assertEqual(rec.num_reverse(test_case), right_answer)
```

```
➜  lab_5 git:(main) ✗ python3 -m unittest -v tests/test_recursions.py
test_fibonacci (tests.test_recursions.TestRecursions) ... ok
test_pow_n (tests.test_recursions.TestRecursions) ... ok
test_recursion_max (tests.test_recursions.TestRecursions) ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```

## Задания на замыкания

- 3. Напишите функцию closure_str, на вход которой подается строка. Функция должна возвращать другую функцию, принимающую номер индекса и возвращающую символ, располагаемый в строке по этому индексу. Если задаваемый индекс выходит за пределы строки, то верните пустой символ.
- 9. Напишите функцию closure_list_pow, на вход которой подается список целочисленных или вещественных значений. Используя механизм замыкания верните функцию, принимающую значение степени, в которую необходимо возвести каждый элемент списка и возвращающую полученный результат.

- 10. Напишите функцию closure_list_del, на вход которой подается список целочисленных значений. Используя механизм замыкания верните функцию, принимающую на вход значение n и возвращающую список, в котором удалены все элементы, что без остатка делятся на n.

```
from typing import List, Union, Callable


Digit = Union[int, float]

def closure_str(string: str) -> Callable[[int], str]:
    '''
    returns a funciton that returns the letter by index
    '''
    def str_index(index: int) -> str:
        if 0 <= index < len(string):
            return string[index]
        return ''
    return str_index

def closure_list_pow(array: List[Digit]) -> Callable[[Digit],
List[Digit]]:
    def list_pow(pow_step: Digit) -> List[Digit]:
        return [el ** pow_step for el in array]
    return list_pow

def closure_list_del(array: List[int]) -> Callable[[int], List[int]]:
    def list_del(divisor: int) -> List[int]:
        return [el for el in array if el % divisor != 0]
    return list_del
```

```
import unittest
import closures as cl


class TestClosures(unittest.TestCase):
    def test_task_3(self):
        '''
        closure_str return function
        '''
        test_string = '012345'
        test_cases = [
            (0, '0'),
            (1, '1'),
            (2, '2'),
            (3, '3'),
            (4, '4'),
            (6, ''),
```

```
            (-1, ''),
        ]
        index_func = cl.closure_str(test_string)
        for test in test_cases:
            param, right_answer = test
            self.assertEqual(index_func(param), right_answer)


    def test_task_9(self):
        '''
        test closure_list_pow
        '''
        test_array = [4, 9, 16]
        test_cases = [
            (1, [4, 9, 16]),
            (0, [1, 1, 1]),
            (2, [16, 81, 256]),
            (0.5, [2, 3, 4])
        ]
        pow_func = cl.closure_list_pow(test_array)
        for test in test_cases:
            param, expected_value = test
            resulting_value = pow_func(param)
            self.assertEqual(resulting_value, expected_value)

    def test_task_10(self):
        test_array = list(range(10))
        test_cases = [
            (1, []),
            (2, [1, 3, 5, 7, 9]),
            (3, [1, 2, 4, 5, 7, 8]),
            (4, [1, 2, 3, 5, 6, 7, 9]),
            (11, list(range(1, 10))),
        ]
        del_func = cl.closure_list_del(test_array)
        for test in test_cases:
            param, expected_value = test
            resulting_value = del_func(param)
            self.assertEqual(resulting_value, expected_value)
```

```
➜  lab_5 git:(main) ✗ python3 -m unittest -v tests/test_closure.py
test_task_10 (tests.test_closure.TestClosures) ... ok
test_task_3 (tests.test_closure.TestClosures)
closure_str return function ... ok
test_task_9 (tests.test_closure.TestClosures)
test closure_list_pow ... ok

----------------------------------------------------------------------
Ran 3 tests in 0.001s
```

```
OK
```