ИУ5-31Б Сигал Д.Э.

Отчет рк2

Программа рк1:

```
from operator import itemgetter

class Composition:
    def __init__(self, id, name, duration, o_id):
        self.id = id
        self.name = name
        self.duration = duration
        self.o_id = o_id

class Orchestra:
    def __init__(self, id, name):
        self.id = id
        self.name = name

class ComOrc:
    def __init__(self, orc_id, com_id):
        self.orc_id = orc_id
        self.com_id = com_id

orchestras = [
    Orchestra(1, 'Оркестр Мариинского театра'),
    Orchestra(2, 'Российский национальный оркестр'),
    Orchestra(3, 'Bavarian Radio Symphony'),

    Orchestra(11, 'Vienna Philharmonic Orchestra'),
    Orchestra(22, 'Czech Philharmonic'),
    Orchestra(33, 'Royal Concertgebouw Orchestra'),
]

compositions = [
    Composition(1, 'Лунная соната', 4.4, 1),
    Composition(2, 'Симфония Бетховена', 12.2, 2),
    Composition(3, 'Симфония Баха', 3, 3),
    Composition(4, 'Симфония Моцарта', 11, 3),
    Composition(5, 'Симфония Сальери', 5.5, 3),
]

comorcs = [
    ComOrc(1,1),
    ComOrc(2,2),
    ComOrc(3,3),
    ComOrc(3,4),
```

```
    ComOrc(3,5),

    ComOrc(11,1),
    ComOrc(22,2),
    ComOrc(33,3),
    ComOrc(33,4),
    ComOrc(33,5),
]
def one_to_many(orchestras,compositions):
  return [(c.name, c.duration, o.name)
        for o in orchestras
        for c in compositions
        if c.o_id==o.id]
def many_to_many(orchestras,compositions):
    many_to_many_temp = [(d.name, co.orc_id, co.com_id)
        for d in orchestras
        for co in comorcs
        if d.id==co.orc_id]

    return [(c.name, c.duration, or_name)
        for or_name, orc_id, com_id in many_to_many_temp
        for c in compositions if c.id==com_id]
def A1(orchestras,compositions) -> list:
    res_11 = sorted(one_to_many(orchestras,compositions),
key=itemgetter(2))
    return(res_11)
def A2(orchestras,compositions) -> list:
    res_12_unsorted = []
    # Перебираем все оркестры
    for o in orchestras:
        # Список произведений оркестра
        o_com = list(filter(lambda i: i[2]==o.name,
one_to_many(orchestras,compositions)))
        # Если у оркестра есть произведения
        if len(o_com) > 0:
            # Длительность произведения
            o_durations = [duration for _,duration,_ in o_com]
            # Суммарная длительность произведений
            o_durations_sum = sum(o_durations)
            res_12_unsorted.append((o.name, o_durations_sum))

    # Сортировка по суммарной длительности произведений
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return(res_12)
def A3(orchestras,compositions,str_to_find) -> list:
    res_13 = {}
    # Перебираем все оркестры
    for o in orchestras:
```

```
        if str_to_find in o.name:
            # Список произведений оркестра
            o_com = list(filter(lambda i: i[2]==o.name,
many_to_many(orchestras,compositions)))
            # Только названия произведений
            o_com_names = [x for x,_,_ in o_com]
            # Добавляем результат в словарь
            # ключ - оркестр, значение - список произведений
            res_13[o.name] = o_com_names

    return(res_13)

if __name__ == '__main__':
    print('Задание А1')
    print(A1(orchestras, compositions))
    print('Задание А2')
    print(A2(orchestras, compositions))
    print('Задание А3')
    print(A3(orchestras, compositions, 'оркестр'))
```

Текст рк2

```
import unittest
from rk1re import Composition, Orchestra, ComOrc, A1, A2, A3

class test(unittest.TestCase):

    def setUp(self):
        self.streets = [
            Orchestra(1, 'Оркестр Мариинского театра'),
            Orchestra(2, 'Российский национальный оркестр'),
            Orchestra(3, 'Bavarian Radio Symphony'),

            Orchestra(11, 'Vienna Philharmonic Orchestra'),
            Orchestra(22, 'Czech Philharmonic'),
            Orchestra(33, 'Royal Concertgebouw Orchestra'),
        ]
        self.houses = [
            Composition(1, 'Лунная соната', 4.4, 1),
            Composition(2, 'Симфония Бетховена', 12.2, 2),
            Composition(3, 'Симфония Баха', 3, 3),
            Composition(4, 'Симфония Моцарта', 11, 3),
            Composition(5, 'Симфония Сальери', 5.5, 3),
        ]
        self.houses_streets = [
              ComOrc(1,1),
              ComOrc(2,2),
```

```
            ComOrc(3,3),
            ComOrc(3,4),
            ComOrc(3,5),

            ComOrc(11,1),
            ComOrc(22,2),
            ComOrc(33,3),
            ComOrc(33,4),
            ComOrc(33,5),
        ]


    def test_A1(self):
        expected_result = [
            ('Симфония Баха', 3, 'Bavarian Radio Symphony'),
            ('Симфония Моцарта', 11, 'Bavarian Radio Symphony'),
            ('Симфония Сальери', 5.5, 'Bavarian Radio Symphony'),
            ('Лунная соната', 4.4, 'Оркестр Мариинского театра'),
            ('Симфония Бетховена', 12.2, 'Российский национальный
оркестр')
        ]

        result = A1(self.streets, self.houses)
        self.assertEqual(result, expected_result)

    def test_A2(self):
        expected_result = [
            ('Bavarian Radio Symphony', 19.5),
            ('Российский национальный оркестр', 12.2),
            ('Оркестр Мариинского театра', 4.4)
        ]
        result = A2(self.streets, self.houses)
        self.assertEqual(result, expected_result)

    def test_A3(self):
        expected_result = {'Российский национальный оркестр': ['Симфония
Бетховена']}
        result = A3(self.streets, self.houses, 'оркестр')
        self.assertEqual(result, expected_result)

if __name__ == '__main__':
    unittest.main()
```

Результат:

```
Ran 3 tests in 0.000s

OK
```