

Измененный текст программы:

```
# используется для сортировки
from operator import itemgetter
# Emp
class House:
    """Дом"""
    def __init__(self, id, number, flats_count, Street_id):
        self.id = id
        self.number = number
        self.flats_count = flats_count
        self.Street_id = Street_id
# Dep
class Street:
    """Улица"""
    def __init__(self, id, name):
        self.id = id
        self.name = name
# EmpDep
class HouseStreet:
    """
    'Дома на улице' для реализации
    связи многие-ко-многим
    """
    def __init__(self, Street_id, House_id):
        self.Street_id = Street_id
        self.House_id = House_id

# Улицы
Streets = [
    Street(1, 'Новый Арбат'),
    Street(2, 'Тверская'),
    Street(3, 'Воздвиженка'),

    Street(11, 'Старый Арбат'),
    Street(22, 'Волхонка'),
    Street(33, 'Большая Никитская'),
]

# Дома
Houses = [
    House(1, 3, 64, 1),
    House(2, 4, 120, 2),
    House(3, 25, 40, 3),
    House(4, 24, 50, 3),
    House(5, 10, 100, 3),
]
```

```

Houses_Streets = [
    HouseStreet(1,1),
    HouseStreet(2,2),
    HouseStreet(3,3),
    HouseStreet(3,4),
    HouseStreet(3,5),

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

def one_to_many():
    return [(e.number, e.flats_count, d.name)
            for d in Streets
            for e in Houses
            if e.Street_id==d.id]

# Соединение данных многие-ко-многим
def many_to_many_temp():
    return [(d.name, ed.Street_id, ed.House_id)
            for d in Streets
            for ed in Houses_Streets
            if d.id==ed.Street_id]

def many_to_many():
    return [(e.number, e.flats_count, Street_name)
            for Street_name, Street_id, House_id in many_to_many_temp()
            for e in Houses if e.id==House_id]

def B1():
    res_11 = sorted(one_to_many(), key=itemgetter(1))
    return (res_11)

def B2():
    res_12_unsorted = []
    # Перебираем все улицы
    for d in Streets:
        # Список домов на улице
        d_Houses = list(filter(lambda i: i[2]==d.name, one_to_many()))
        house_count=len(d_Houses)
        res_12_unsorted.append((d.name, house_count))
    # Сортировка по кол-ву домов
    res_12 = sorted(res_12_unsorted, key=itemgetter(1), reverse=True)
    return(res_12)

def B3():
    res_13 = {}
    # Перебираем все улицы
    for d in Houses:
        if d.number>10:
            d_Streets = list(filter(lambda i: i[0]==d.number, many_to_many()))
            d_Streets_names=[x for _,x in d_Streets]
            res_13[d.number]=d_Streets_names
    return(res_13)

def main():
    print('Задание Б1 (по кол-ву квартир)')
    print(B1())
    print('\nЗадание Б2')
    print(B2())
    print('\nЗадание Б3 (если номер дома больше 10)')
    print(B3())

if __name__ == '__main__':
    main()

```

Тестирование:

```
import unittest
from BKIT_RK1 import *

class test_rk(unittest.TestCase):
    def set(self):
        self.Streets = [
            Street(1, 'Новый Арбат'),
            Street(2, 'Тверская'),
            Street(3, 'Воздвиженка'),

            Street(11, 'Старый Арбат'),
            Street(22, 'Волхонка'),
            Street(33, 'Большая Никитская'),
        ]
        self.Houses = [
            House(1, 3, 64, 1),
            House(2, 4, 120, 2),
            House(3, 25, 40, 3),
            House(4, 24, 50, 3),
            House(5, 10, 100, 3),
        ]
        self.Houses_Streets = [
            HouseStreet(1,1),
            HouseStreet(2,2),
            HouseStreet(3,3),
            HouseStreet(3,4),
            HouseStreet(3,5),

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

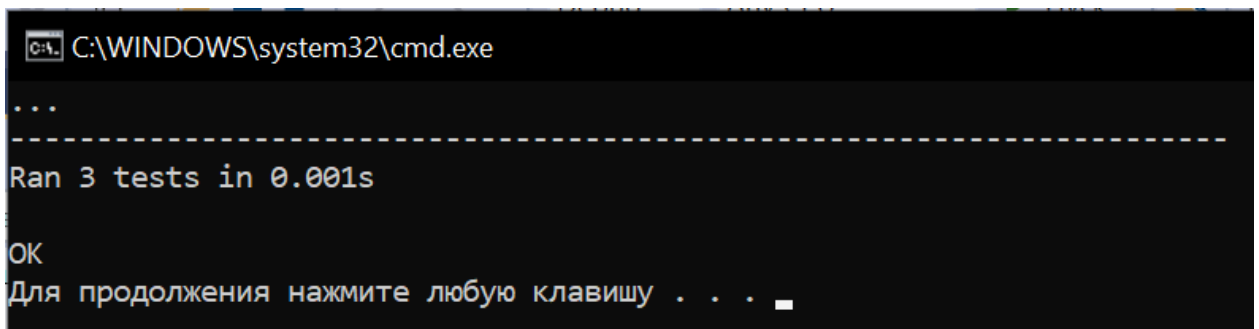
    def test_B1(self):
        expected_res = [(25, 40, 'Воздвиженка'), (24, 50, 'Воздвиженка'),
                        (3, 64, 'Новый Арбат'), (10, 100, 'Воздвиженка'), (4, 120, 'Тверская')]
        res = B1()
        self.assertEqual(res, expected_res)

    def test_B2(self):
        expected_res = [('Воздвиженка', 3), ('Новый Арбат', 1),
                        ('Тверская', 1), ('Старый Арбат', 0), ('Волхонка', 0), ('Большая Никитская', 0)]
        res = B2()
        self.assertEqual(res, expected_res)

    def test_B3(self):
        expected_res = {25: ['Воздвиженка', 'Большая Никитская'], 24: ['Воздвиженка', 'Волхонка']}
        res = B3()
        self.assertEqual(res, expected_res)

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

Результат работы программы:



```
C:\WINDOWS\system32\cmd.exe

...

-----
Ran 3 tests in 0.001s
ОК
Для продолжения нажмите любую клавишу . . .
```