Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

Факультет «Информатика и системы управления»

ДИСЦИПЛИНА:

«БКИТ»

Рубежная контроль № 2

Студент Распашнов А.А. ИУ5Ц-54Б _____

(И.О. Фамилия) (Группа) (Подпись, дата)

Преподаватель Гапанюк Ю.Е. _____

(И.О. Фамилия) (Подпись, дата)

Условия рубежного контроля №2 по курсу БКИТ

Рубежный контроль представляет собой разработку тестов на языке Python.

1) Проведите рефакторинг текста программы рубежного контроля №1 таким образом, чтобы он был пригоден для модульного тестирования.

2) Для текста программы рубежного контроля №1 создайте модульные тесты с применением TDD - фреймворка (3 теста).

Main.py:

```
import unittest


class Oper:
    def __init__(self, id, name, salary, lang_id):
        self.id = id
        self.name = name
        self.salary = salary
        self.lang_id = lang_id


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


class OperLang:
    def __init__(self, oper_id, lang_id):
        self.oper_id = oper_id
        self.lang_id = lang_id


languages = [
    Lang(1, 'Python'),
    Lang(2, 'Java'),
    Lang(3, 'AspectJ'),
    Lang(4, 'C++'),
    Lang(5, 'C'),
    Lang(6, 'Go'),
    Lang(7, 'Ada')
]

operators = [
    Oper(1, 'Maxim', 1000, 1),
    Oper(2, 'Andrey', 2000, 2),
    Oper(3, 'Daniel', 1500, 1),
    Oper(4, 'Viktor', 3000, 5),
    Oper(5, 'Misha', 2500, 4),
    Oper(6, 'Dima', 500, 7),
```

```
    Oper(7, 'Rebekka', 300, 3)
]

opers_langs = [
    OperLang(1, 1),
    OperLang(1, 5),
    OperLang(1, 3),
    OperLang(2, 2),
    OperLang(2, 3),
    OperLang(3, 1),
    OperLang(4, 2),
    OperLang(4, 3),
    OperLang(6, 1),
    OperLang(6, 5)
]


def filter_letter(one_to_many, letter):
    return [res for res in filter(lambda entry: entry[2][0] == letter,
one_to_many)]


def sorted_max_salary(one_to_many):
    return sorted([(lang.name, max([salary for _, salary, lang_name in
one_to_many if lang_name == lang.name]))
                                for lang in languages if len(list(filter(lambda
entry: entry[2] == lang.name, one_to_many))) > 0],
                               key=lambda x: x[1],
                               reverse=True)


def links_sorted_langs(many_to_many):
    return sorted(many_to_many, key=lambda entry: entry[2])


def main():
    one_to_many = [(oper.name, oper.salary, lang.name)
                                for lang in languages
                                for oper in operators
                                if oper.lang_id == lang.id]

    many_to_many_temp = [(lang.name, ol.oper_id, ol.lang_id)
                                            for lang in languages
                                            for ol in opers_langs
                                            if lang.id == ol.lang_id]

    many_to_many = [(oper.name, oper.salary, lang_name)
                                    for lang_name, _, lang_id in
many_to_many_temp
                                    for oper in operators if oper.lang_id ==
lang_id]
```

```
    print('Задание Г1')
    print(filter_letter(one_to_many, 'A'))

    print('\nЗадание Г2')
    print(sorted_max_salary(one_to_many))

    print('\nЗадание Г3')
    print(links_sorted_langs(many_to_many))


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

test.py

```
import unittest

from main import *

one_to_many = [(oper.name, oper.salary, lang.name)
                                for lang in languages
                                for oper in operators
                                if oper.lang_id == lang.id]

many_to_many_temp = [(lang.name, ol.oper_id, ol.lang_id)
                                        for lang in languages
                                        for ol in opers_langs
                                        if lang.id == ol.lang_id]

many_to_many = [(oper.name, oper.salary, lang_name)
                                for lang_name, _, lang_id in many_to_many_temp
                                for oper in operators if oper.lang_id == lang_id]


class Tests(unittest.TestCase):
    def test_filter_letter(self):
        self.assertEqual(filter_letter(one_to_many, 'A'), [('Rebekka', 300,
'AspectJ'), ('Dima', 500, 'Ada')])

    def test_sorted_max_salary(self):
        self.assertEqual(sorted_max_salary(one_to_many), [('C', 3000), ('C++',
2500), ('Java', 2000), ('Python', 1500),

                          ('Ada', 500), ('AspectJ', 300)])

    def test_links_sorted_langs(self):
        self.assertEqual(links_sorted_langs(many_to_many), [('Rebekka', 300,
'AspectJ'), ('Rebekka', 300, 'AspectJ'),
```

```
                        ('Rebekka', 300, 'AspectJ'), ('Viktor', 3000, 'C'),
                        ('Viktor', 3000, 'C'), ('Andrey', 2000, 'Java'),
                        ('Andrey', 2000, 'Java'), ('Maxim', 1000, 'Python'),
                        ('Daniel', 1500, 'Python'), ('Maxim', 1000, 'Python'),
                        ('Daniel', 1500, 'Python'), ('Maxim', 1000, 'Python'),
                        ('Daniel', 1500, 'Python')])

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

Результат: