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»
Кафедра «Системы обработки информации и управления»

Новицкий Ярослав ИУ5-35Б

Парадигмы и конструкции языков программирования

ОТЧЁТ ПО
Рубежному контролю №2

Москва
2023

Текст программы:

main.py

```
import re


class Book:
    def __init__(self, id_number, name, author, year, lib_id):
        self.id_number = id_number
        self.name = name
        self.author = author
        self.year = year
        self.lib_id = lib_id


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


class BookLib:
    def __init__(self, lib_id, book_id):
        self.lib_id = lib_id
        self.book_id = book_id


libs = [
    Lib(1, "Central Library"),
    Lib(2, "City Public Library"),
    Lib(3, "University Library"),
    Lib(4, "Community Library"),
    Lib(5, "School Library")
]


books = [
    Book(1, "To Kill a Mockingbird", "Harper Lee", 1960, 1),
    Book(2, "1984", "George Orwell", 1949, 2),
    Book(3, "Pride and Prejudice", "Jane Austen", 1813, 3),
    Book(4, "The Great Gatsby", "F. Scott Fitzgerald", 1925, 1),
    Book(5, "Brave New World", "Aldous Huxley", 1932, 2),
    Book(6, "The Catcher in the Rye", "J.D. Salinger", 1951, 1),
    Book(7, "The Little Prince", "Antoine de Saint-Exupéry", 1943, 3),
    Book(8, "The Lord of the Rings", "J.R.R. Tolkien", 1954, 4),
    Book(9, "The Hobbit", "J.R.R. Tolkien", 1937, 4),
    Book(10, "War and Peace", "Leo Tolstoy", 1869, 5)
]

book_lib = [
    BookLib(1, 1),
    BookLib(2, 2),
    BookLib(3, 3),
    BookLib(1, 4),
    BookLib(2, 5),

    BookLib(1, 6),
    BookLib(3, 7),
    BookLib(4, 8),
    BookLib(4, 9),
```

```
    BookLib(5, 10),
]


def main():

    # Соединение данных один-ко-многим
    one_to_many = [(b.name, b.year, ll.name)
                   for ll in libs
                   for b in books
                   if b.lib_id == ll.id]

    # Соединение данных многие-ко-многим
    many_to_many_temp = [(ll.name, bl.lib_id, bl.book_id)
                         for ll in libs
                         for bl in book_lib
                         if ll.id == bl.lib_id]

    many_to_many = [(b.name, b.year, b.author, lib_name)
                    for lib_name, lib_id, book_id in many_to_many_temp
                    for b in books if b.id_number == book_id]

    print('Задание Д1')
    res_11 = []
    for book_name, year, lib_name in one_to_many:
        matches = re.findall(r'\b\w+ce\b', book_name)
        if matches:
            res_11.append((book_name, lib_name))
    print(res_11)
    # средний год написания книги в библиотеке
    print('\nЗадание Д2')
    res_12 = {}
    for ll in libs:
        l_books = list(filter(lambda i: i[2] == ll.name, one_to_many))
        if len(l_books) > 0:
            l_books_years = [x for _, x, _ in l_books]
            res_12[ll.name] = int(sum(l_books_years)/len(l_books_years))
    print(sorted(res_12.items(), key=lambda item: item[1]))
    print('\nЗадание Д3')
    res_13 = {}
    for ll in libs:
        if ll.name[0] == 'C':
            l_books = list(filter(lambda i: i[3] == ll.name, many_to_many))
            l_books_names = [x for x, _, _, _ in l_books]
            res_13[ll.name] = l_books_names
    print(res_13)


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

tests.py

```
from io import StringIO
import unittest
from unittest.mock import patch, mock_open
import main


class TestProgram(unittest.TestCase):
```

```
    @patch("builtins.open", mock_open(read_data="test data"))
    def test_filter_books_ending_with_ce(self):
        expected_result = [('Pride and Prejudice', 'University Library'), ('The
Little Prince', 'University Library'), ('War and Peace', 'School Library')]
        with patch("sys.stdout", new_callable=StringIO) as mock_stdout:
            main.main()
            actual_output = mock_stdout.getvalue()
            self.assertTrue(all(str(book) in actual_output for book in ex-
pected_result))

    @patch("builtins.open", mock_open(read_data="test data"))
    def test_average_year_by_lib(self):
        expected_result = [('School Library', 1869), ('University Library',
1878), ('City Public Library', 1940), ('Central Library', 1945), ('Community Li-
brary', 1945)]
        with patch("sys.stdout", new_callable=StringIO) as mock_stdout:
            main.main()
            actual_output = mock_stdout.getvalue()
            self.assertTrue(all(str(year) in actual_output for year in ex-
pected_result))

    @patch("builtins.open", mock_open(read_data="test data"))
    def test_filter_libs_starting_with_c(self):
        expected_result = {'Central Library': ['To Kill a Mockingbird', 'The
Great Gatsby', 'The Catcher in the Rye'], 'City Public Library': ['1984', 'Brave
New World'], 'Community Library': ['The Lord of the Rings', 'The Hobbit']}
        with patch("sys.stdout", new_callable=StringIO) as mock_stdout:
            main.main()
            actual_output = mock_stdout.getvalue()
            self.assertTrue(all(Lib in actual_output for Lib in expected_re-
sult.keys()))
            self.assertTrue(all(all(Book in actual_output for Book in Books) for
Lib, Books in expected_result.items()))

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

Результаты выполнения: