Практические задачи по курсу ООП

Задания выполняются на языке Java. Все решения должны сопровождаться набором тестов, проверяющим корректность выполненного задания. Сдаваемая задача должна компилироваться с командной строки командой «gradle build».

1. Знакомство с языком программирования Java

1.1 Пирамидальная сортировка `Task_1_1_1`

Реализуйте классический алгоритм пирамидальной сортировки.
Кроме настроенной сборки через gradle, необходимо предоставить shell-скрипт (без использования системы сборки), который компилирует исходники, генерирует документацию и запускает приложение.

Вход	`heapsort(new int[] {5, 4, 3, 2, 1});`
Выход	`[1, 2, 3, 4, 5]`

1.2 Операции с многочленами `Task_1_1_2`

Реализуйте объектное представление многочлена степени n.

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0, \quad a_i \in R,\ a_n \neq 0$$

Класс должен поддерживать операции:

- создание многочлена с заданными коэффициентами;
- сложение, вычитание, умножение на другой многочлен;
- вычисление значения в точке;
- взятие i'ой производной;
- сравнение на равенство с другим многочленом;
- получение строкового представления.

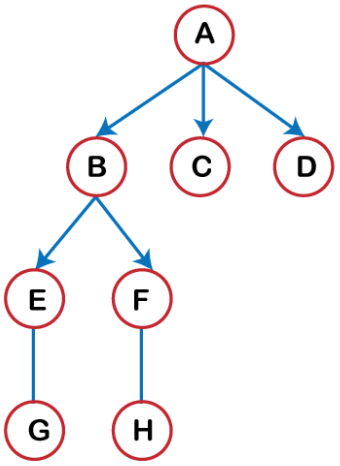
В реализации запрещено использовать стандартные контейнеры языка Java, кроме массива.

Вход	`Polynomial p1 = new Polynomial(new int[] {4, 3, 6, 7});`
	`Polynomial p2 = new Polynomial(new int[] {3, 2, 8});`
	`System.out.println(p1.plus(p2.differentiate(1)).toString());`
	`System.out.println(p1.times(p2).evaluate(2));`

Выход	7x^3 + 6x^2 + 19x + 6 3510

2. Контейнеры

2.1 Дерево `Task_1_2_1`



Реализуйте generic-класс для классического дерева — связного ациклического ориентированного графа. Дерево должно поддерживать:

- добавление и удаление элементов/поддеревьев;
- стандартный механизм итерирования элементов коллекции методом обхода в ширину и глубину с обработкой исключительной ситуации ConcurrentModificationException;
- сравнение на равенство с другим деревом.

В качестве дополнительного задания покажите, как можно осуществлять поиск по дереву с использованием Stream API. Изучите понятие *сплитератора*. Поделитесь рассуждениями с преподавателем.

Вход	`// Интерфейс вашего класса Tree может отличаться`
	`Tree<String> tree = new Tree<>("R1");`
	`var a = tree.addChild("A");`
	`var b = a.addChild("B");`
	`Tree<String> subtree = new Tree<>("R2");`
	`subtree.addChild("C");`
	`subtree.addChild("D");`
	`tree.addChild(subtree);`
	`b.remove();`

Выход	`R1`
	`/ \`
	`A R2`
	`/ \`
	`C D`

2.2 Граф `Task_1_2_2`

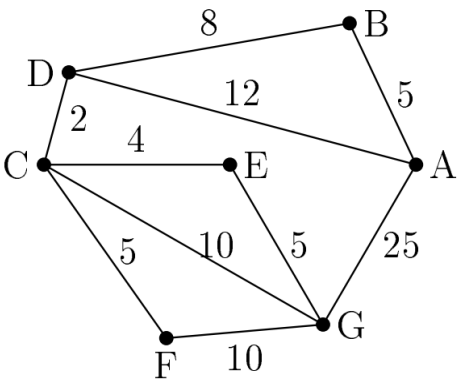
Реализуйте библиотеку для работы с ориентированными взвешенными графами, поддерживающую 3 классических способа представления: через матрицу смежности, матрицу инцидентности и списки смежности. Объектная модель должна содержать, как минимум, граф, вершину и ребро. Объект параметризованного класса Graph должен поддерживать операции:

- добавить/удалить вершину;
- добавить/удалить ребро;
- получить/изменить объект, соответствующий вершине или ребру;
- другие необходимые операции.

Для созданного объекта Graph реализуйте алгоритм сортировки вершин по удалённости от заданной вершины. При сдаче задания преподавателю необходимо рассказать, как добавить еще один способ представления графов (например, в виде “структуры с оглавлением”) и новый алгоритм (например, поиск кратчайшего расстояния между вершинами). Качественно сравните три реализованных способа представления графов для алгоритма сортировки.

Для тестирования загружайте начальное состояние графов из текстового файла.

В качестве дополнительного задания добавьте новый алгоритм, решающий ту же задачу и допускающий графы с отрицательно взвешенными ребрами.

Задача	`Для городов A, B, C, D, E, F, G имеется сеть дорог, каждая дорога характеризуется расстоянием и типом покрытия, из которых можно вычислить требуемое время в пути между городами. Вывести в отсортированном по временной удаленности населенные пункты от города C.` 

<table>
<tr><td>Вход</td><td>input.txt<table>
<tr><td></td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td><td>G</td></tr>
<tr><td>A</td><td></td><td>5</td><td></td><td>12</td><td></td><td></td><td>25</td></tr>
<tr><td>B</td><td>5</td><td></td><td></td><td>8</td><td></td><td></td><td></td></tr>
<tr><td>C</td><td></td><td></td><td></td><td>2</td><td>4</td><td>5</td><td>10</td></tr>
<tr><td>D</td><td>12</td><td>8</td><td>2</td><td></td><td></td><td></td><td></td></tr>
<tr><td>E</td><td></td><td></td><td>4</td><td></td><td></td><td></td><td>5</td></tr>
<tr><td>F</td><td></td><td></td><td>5</td><td></td><td></td><td></td><td>5</td></tr>
<tr><td>G</td><td>25</td><td></td><td>10</td><td></td><td>5</td><td>5</td><td></td></tr>
</table></td></tr>
<tr><td>Выход</td><td>[C(0), D(2), E(4), F(5), G(9), B(10), A(14)]</td></tr>
</table>

3. Ввод/вывод

3.1 Поиск подстроки `Task_1_3_1`

На вход подаётся имя файла и строка, которую надо найти. Строка может содержать буквы любого алфавита в кодировке UTF-8. Реализуйте функцию, определяющую индекс начала каждого вхождения заданной подстроки.

Размер входного файла может быть во много раз больше объема оперативной памяти вычислительного устройства, а размер искомой подстроки — во много раз меньше.

Файл	`input.txt:` `абракадабра`
Вход	`find("input.txt", "бра");`
Выход	`[1, 8]`

4. Типы данных

4.1 Зачетная книжка `Task_1_4_1`

Реализуйте класс электронной зачетной книжки студента ФИТ, обеспечьте функции вычисления:

- текущего среднего балла за все время обучения;
- возможности получения «красного» диплома с отличием;
- возможности получения повышенной стипендии в этом семестре.

Требования для диплома с отличием: 75% оценок в приложении к диплому (последняя оценка) – “отлично”, отсутствие итоговых оценок “удовлетворительно” и квалификационная работа на “отлично”.

5. Консольные приложения

5.1 Калькулятор `Task_1_5_1`

Реализуйте калькулятор инженера для вещественных чисел. Пользователь вводит на стандартный поток ввода выражение в префиксном виде, калькулятор вычисляет значение и выводит на стандартный поток вывода. Кроме стандартных операций (+, -, *, /), поддержите набор функций (log, pow, sqrt, sin, cos).

Предложенное решение должно поддерживать разные типы возможных исключительных ситуаций.

В качестве дополнительного задания добавьте поддержку градусов и комплексных чисел.

Вход	`sin + - 1 2 1`
Выход	`0`

5.2 Записная книжка `Task_1_5_2`

Реализуйте класс записной книжки с набором функций, доступных с командной строки:

- добавить/удалить запись;
- вывести все записи, отсортированные по времени добавления;
- вывести записи, отсортированные по времени добавления из заданного интервала времени и содержащие в заголовке ключевые слова.

Данные записной книжки должны сериализоваться в файл формата JSON. Для работы с JSON используйте библиотеки Jackson или Gson. Для анализа параметров командной строки так же используйте стороннюю библиотеку (любую понравившуюся вам).

Вход	`notebook -add “Моя заметка” “Очень важная заметка”`
	`notebook -rm “Моя заметка”`
	`notebook -show`
	`notebook -show “14.12.2019 7:00” “17.12.2019 13:00” “МоЯ” “Твоя” “мне”`