Обязательные Задачи.

о всех условиях заданий подразумевается, что вводные данные - это отдельный файл, и вывод программы - это тоже отдельных файл.

№1 Написать Qsort без рекурсии (1 балл).

Требуется реализовать QSort без рекурсии, с использованием стека.
Использование рекурсии в этом задание считается обманом.
На вход подается один файл (**in.txt**), в котором содержится строка не отсортированных чисел, ваш скрипт должен записать в другой файл (**out.txt**) результат сортировки.

Input file	**Output file**
-6 -3 6 4 -10 -1	-10 -6 -3 -1 4 6
-1 3 5 3 -2 -7	-7 -2 -1 3 3 5

Таблица 1. Иллюстрация примера ввода и вывода. Каждая строка - отдельный случай.

№2 Разбор формулы без приоритета (1.5 балл).

Требуется реализовать разбор формулы без приоритета. Иначе говоря, разобрать формулу считая приоритетными операторы встретившиеся раньше по порядку. Например, выражение 2 + 2 * 2 будет равно восьми. Но приоритет задаваемых скобками сохраняется (2 + (2 * 2) уже будет равно 6, как и должно).
На вход подается один файл (**in.txt**), в котором содержится формула или выражения, ваш скрипт должен записать в другой файл (**out.txt**) результат вычисления.

Input file	**Output file**
2 + 2 * 2	8
10 + 9 * (14 * 20) (5 + 6 + 6) / (5-5) 13 + 17 * 19 + 20	5320 ZERODIVISION 590

№3 Разбор формулы с приоритетом (2 балла).

Требуется реализовать разбор формулы с приоритетом. В выражениях присутствуют операции +, -, / и *. Приоритет умножения и деления над сложением и вычитанием есть.
На вход подается один файл (**in.txt**), в котором содержится одно или несколько выражений, ваш скрипт должен записать в другой файл (**out.txt**) результат вычисления.

Input file	**Output file**
2 + 2 * 2	6
10 + 9 * (14 * 20) (5 + 6 + 6) / (5-5) 13 + 17 * 19 + 20	2530 ZERODIVISION 356

№4 Бинарное дерево поиска (1 балл).

Требуется реализовать бинарное дерево поиска.
Должно выполняться следующее:

> **Если в дерево вставляется значение, которое там уже есть, то ничего делать не надо (считаем, что это значение мы уже вставили)**

На вход подается один файл (**in.txt**), в котором содержится строка из разделенных пробелом чисел, ваш скрипт должен записать в другой файл (**out.txt**) дерево в скобочной форме, построенное последовательным добавлением значений во входной строке от левого к правому.
Скобочная форма:
Узел записывается в формате значение ноды, левый потомок, правый потомок.
Пустые ноды обозначаются пустыми скобками.

Input file	**Output file**	Иллюстрация, в задании подобного делать не требуется
5 6 1	5, (1, (), ()), (6, (), ())	5 / \

		1 6
5 6 1 0 2 7	5, (1, (0, (), ()), (2, (), ())), (6, (), (7, (), ()))	5 / \ 1 6 / \ \ 0 2 7

№5 Бинарное дерево поиска (1.5 балл).

Требуется реализовать бинарное дерево поиска как и в задании №4.
Должно выполняться следующее:

> **Если в дерево вставляется значение, которое там уже есть, то ничего делать не надо (считаем, что это значение мы уже вставили)**

Полностью реализовать процедуры вставки и удаления элемента по значению.

1. **insert число** добавляет числов в дерево, возвращает True. Если число уже есть в дереве, то не добавляет возвращает False.
2. **find число** ищет число в дереве, если есть, возвращает True, если нет, то возвращает False
3. **remove число** ищет число в дереве, если есть, то удаляет его и возвращает True, если нет, то возвращает False
4. **print** выводит дерево как в четвертой задаче.

На вход подается один файл (**in.txt**), в котором содержится список последовательных команд, которые следует последовательно исполнять записывая в выходной файл (**out.txt**) промежуточные результаты действий с деревом.
Подсказка, для использования функций при подаче строки может быть удобно использовать getattr().

Input file	**Output file**
insert 5 insert 3 insert 7 print find 5 remove 5 print	INPUT: insert 5 OUTPUT: True INPUT: insert 3 OUTPUT: True INPUT: insert 7 OUTPUT: True INPUT: print OUTPUT: 5, (3, (), ()), (7, (), ()) INPUT: find 5

	OUTPUT: True INPUT: remove 5 OUTPUT: True INPUT: print OUTPUT: 7, (3, (), ()), ()
insert 1 insert 2 insert 3 print find 2 remove 2 print	INPUT: insert 1 OUTPUT: True INPUT: insert 2 OUTPUT: True INPUT: insert 3 OUTPUT: True INPUT: print OUTPUT: 1, (), (2, (), (3, (), ())) INPUT: find 2 OUTPUT: True INPUT: remove 2 OUTPUT: True INPUT: print OUTPUT: 1, (), (3, (), ())

№6 **Count_height (1 балл)**

Постройте дерево, добавляя в него числа в том порядке, в котором они представлены в входе.

Должно выполняться следующее:

> **Если в дерево вставляется значение, которое там уже есть, то вставляем новое значение в левое поддерево (иначе говоря, для каждого узла выполнено, что слева лежат элементы меньше или равные ему, а справа - строго большие)**

Посчитайте максимальную высоту полученного дерева поиска. Высота считается от 0

In:	Строка, где через пробел записаны числа, по которым строится дерево.
Out:	Натуральное число - высота дерева.

Example:

In:	2
Out:	0

In:	2 4 5 1 3

Out:	2

Дополнительные Задачи.

№7 Разбор формулы с приоритетом (1.5 балла).

Требуется реализовать разбор формулы с приоритетом. В выражениях присутствуют операции +, -, /, * и возведение в степень. Приоритет умножения и деления над сложением и вычитанием есть, и возведения в степень над всеми остальными.
На вход подается один файл (**in.txt**), в котором содержится одно или несколько выражений, ваш скрипт должен записать в другой файл (**out.txt**) результат вычисления.
Подсказка: возведение в степень - правоассоциативная операция. Умножение, сложение, вычитание и деление - левоассоциативные.

Input file	**Output file**
2 + 2 ** 2 * 2	10
5 + 2 * 14 ** 3 (5 + 6 + 6) / (5-5) 7 + 8 ** 9 + 5	5493 ZERODIVISION 134217740

№8 Способы получения одного дерева (1.5 балла).

В задании номер 4, дерево стоило строить из последовательности чисел. В этом задании также подается массив чисел, на базе которого последовательно строится дерево.
Вопрос, сколько существует различных последовательностей чисел, которые в итоге будут порождать одно и то же дерево?
На вход подается один файл (**in.txt**), в котором содержится последовательность чисел строкой, ваш скрипт должен записать в другой файл (**out.txt**) все возможные последовательности для порождения того же дерева,

Input file	Output file
1 2 3 4 5	1 2 3 4 5
3 2 5 6	3 2 5 6 3 5 6 2 3 5 2 6

№9 Рандомизированное дерево поиска (2 балла).

Реализуйте **рандомизированное дерево поиска**
Ссылка на тему - https://neerc.ifmo.ru/wiki/index.php?title=%D0%A0%D0%B0%D0%BD%D0%B4%D0%BE%D0%BC%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B1%D0%B8%D0%BD%D0%B0%D1%80%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B0

Должно выполняться следующее:

> **Если в дерево вставляется значение, которое там уже есть, то ничего делать не надо (считаем, что это значение мы уже вставили)**

Полностью реализовать процедуры вставки и удаления элемента по значению.

1. **insert число** добавляет числов в дерево, возвращает True. Если число уже есть в дереве, то не добавляет возвращает False.
2. **find число** ищет число в дереве, если есть, возвращает True, если нет, то возвращает False
3. **remove число** ищет число в дереве, если есть, то удаляет его и возвращает True, если нет, то возвращает False
4. **print** выводит дерево как в четвертой задаче.

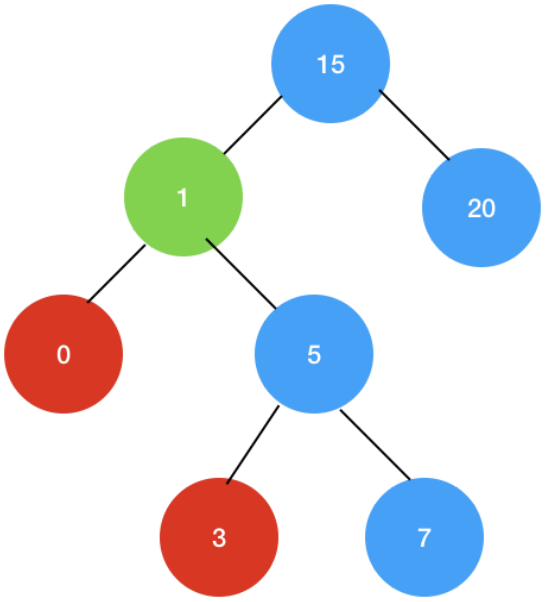
На вход подается один файл (**in.txt**), в котором содержится список последовательных команд, которые следует последовательно исполнять записывая в выходной файл (**out.txt**) промежуточные результаты действий с деревом.
Подсказка, для использования функций при подаче строки может быть удобно использовать getattr().

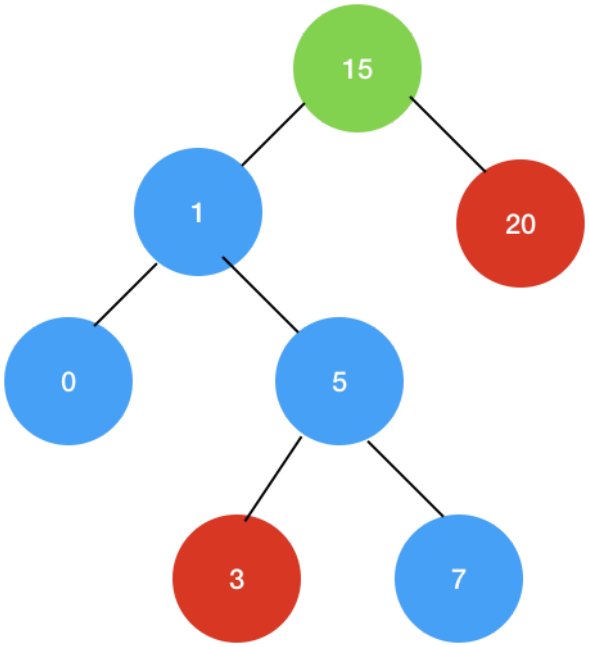
Input file	Output file
insert 5 insert 3 insert 7 print find 5 remove 5 print	INPUT: insert 5 OUTPUT: True INPUT: insert 3 OUTPUT: True INPUT: insert 7 OUTPUT: True INPUT: print OUTPUT: 5, (3, (), ()), (7, (), ()) INPUT: find 5 OUTPUT: True INPUT: remove 5 OUTPUT: True INPUT: print OUTPUT: 7, (3, (), ()), ()
insert 1 insert 2 insert 3 print find 2 remove 2 print	INPUT: insert 1 OUTPUT: True INPUT: insert 2 OUTPUT: True INPUT: insert 3 OUTPUT: True INPUT: print OUTPUT: 1, (), (2, (), (3, (), ())) INPUT: find 2 OUTPUT: True INPUT: remove 2 OUTPUT: True INPUT: print OUTPUT: 1, (), (3, (), ())

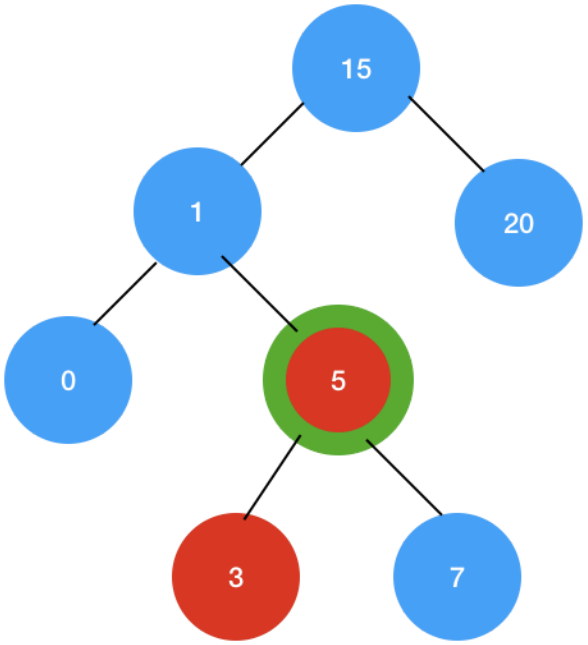
№10 **LCA (2 балла)**

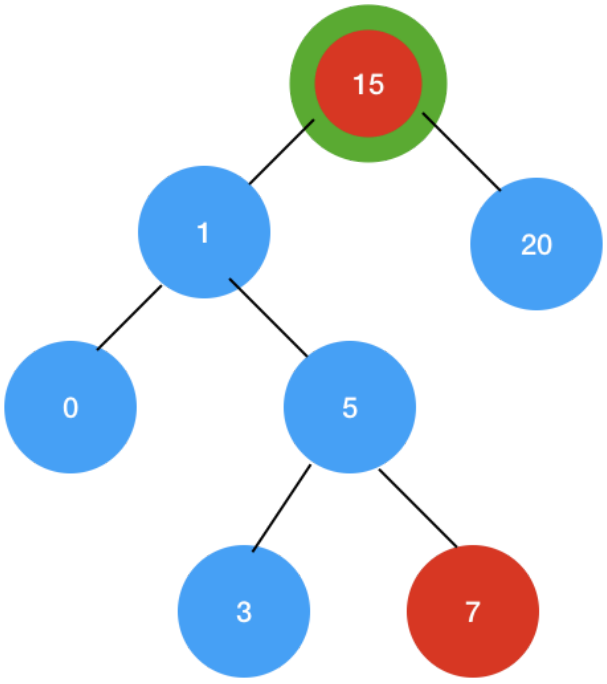
Реализуйте алгоритм поиска наименьшего общего предка. Он должен работать не более чем за O(logH) на запрос, где H - высота вашего дерева.
Используйте именно простое бинарное дерево поиска!
https://en.wikipedia.org/wiki/Lowest_common_ancestor

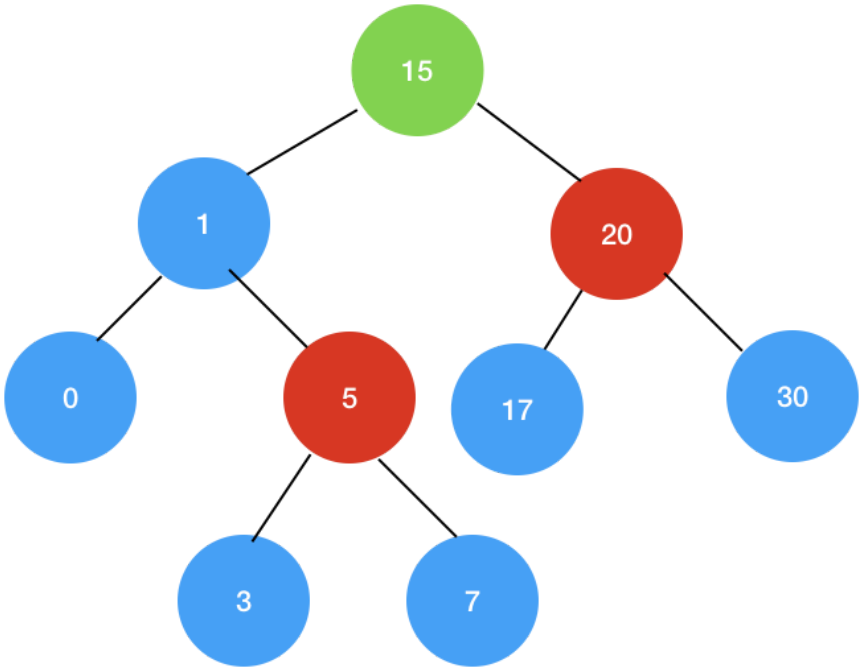
Примеры, зеленым выделен LCA, красным - узлы, для которых мы его ищем


15
1
20
0
5
3
7


15
1
20
0
5
3
7


15
1
20
0
5
3
7


15
1
20
0
5
3
7



In:	В первой строке через пробел записаны уникальные числа, по которым строится исходное дерево. В последующих строках записаны запросы в виде первое_число второе_число. Необходимо вывести их lca.
Out:	Ответы на lca-запросы

In:	15 20 30 17 1 0 5 3 7 0 3 3 20 20 3 3 5 7 15 5 20
Out:	1 15 15 5 15 15