Общие требования

Если в какой-то задаче вам понадобится сбалансированное бинарное дерево поиска, его необходимо реализовывать самим (красно-черное, AVL или декартово дерево).

Кроме того, его необходимо будет протестировать, помимо сдачи задачи в систему. Как минимум каждый публичный метод должен быть протестирован. Как на маленьких ручных тестах, создающих и уничтожающих заранее фиксированные деревья, так и на случайных сгенерированных последовательностях добавлений, поисков и удалений элементов. Не забывайте про крайние тесты, когда в дерево пустое или из одного элемента, когда среди ключей дерева есть крайние значения (границы значений целочисленного типа), когда из дерева удаляется последний элемент и т.д. Каждый тест должен быть реализован прямо в коде в виде функции. Каждый особый случай нужно реализовывать в виде отдельного теста (функции).

Кроме того, нужно тестировать на маленьких тестах выполнение всех обязательных свойств соответствующего дерева после выполнения каждой операции, для этого нужно будет реализовать отдельный метод `CheckStructure` только для тестирования, который будет проверять количество черных вершин на пути от корня до каждого листа, нет ли красных детей у красной вершины, будет сравнивать реальную глубину дерева с теоретической оценкой, проверять, является ли дерево вообще корректным деревом поиска и т.д. и т.п. Естественно, все проверки нужно реализовывать в отдельных методах, и вызывать их из этого общего. Этот метод, скорее всего, будет работать за $O(n)$, где n — количество вершин в дереве на данный момент, поэтому вызывать его нужно только когда тестируете на маленьких тестах: на больших все будет тормозить.

Кроме того, нужно проконтролировать, что не течет память, и для этого нужно при выделении вершины увеличивать какой-нибудь внутренний тестовый счетчик на 1, при удалении — уменьшать, и в конце проверять, что этот счетчик равен нулю.

Почитать о том, что такое юнит тесты можно, например, здесь
`http://en.wikipedia.org/wiki/Unit_testing`
а пример постепенного создания хорошего набора тестов — здесь
`http://www.c2.com/cgi/wiki?CodeUnitTestFirstExampleTwo`

Задача 5-1. Рассмотрим множество отрезков на прямой с целыми концами. Изначально множество пустое, в него могут добавляться отрезки, из него могут удаляться отрезки. После каждой операции вставки или удаления отрезка необходимо вывести общую длину объединения всех отрезков, лежащих на данный момент в множестве.

В первой строке входа записано целое число $1 \le n \le 100\ 000$ — общее количество проделанных операций. Далее идут n строк, каждая из них устроена следующим образом. Первый символ — "+", если это операция вставки отрезка и "-", если это операция удаления отрезка. Далее в строке записаны через пробел два целых числа — левый и правый концы отрезка. Координаты концов по модулю не превосходят 1 000 000 000. Гарантируется, что удаляться будут только отрезки, которые перед этим были добавлены в множество. Выведите ровно n чисел, по одному в строке — общую длину объединения всех отрезков в множестве после каждой из n операций добавления/удаления.

Пример входа	Пример выхода
4 + 1 3 + 2 6 + 5 7 - 2 6	2 5 6 4

Задача 5-2.

Имеется циклическая автомобильная стоянка, то есть такая, в которой все парковочные места распололжены по кругу. Всего есть n парковочных мест, и они пронумерованы от 1 до n. На стоянку приезжают и уезжают автомобили. Каждый автомобиль заранее целенаправленно подъезжает к определенному парковочному месту — месту, выделенному для хозяина автомобиля. Однако частенько случается, что кто-то уже занял это место, тогда автомобиль начинает ехать по часовой стрелке (числа расположены в порядке $1, 2, 3, \ldots, n, 1, 2, 3, \ldots$ по часовой стрелке) до тех пор, пока не найдет первое свободное место и не встанет на него. Если же мест нет, то автомобиль уезжает, так и не попав на стоянку.

В первой строке входа записано два целых числа $1 \leq n, m \leq 100\,000$, где n — количество мест на стоянке, а m — количество событий, произошедших за день. Событие — это либо приезд, либо отъезд автомобиля. В следующих m строках — по событию в строке. Прибытие автомобиля записывается как "+ i", где i — номер места, к которому изначально направляется автомобиль. Отъезд записывается как "- i", где i — номер места на стоянке.

Для каждого приезда автомобиля выведите единственную строку, в которой записан номер места, на которое он встал, либо -1, если места не нашлось. Для каждого отъезда автомобиля выведите 0 если на данном месте действительно был автомобиль и -2 если автомобиля на месте i не было.

Пример входа	Пример выхода
4 8 + 2 + 2 + 2 + 2 + 2 + 2 - 2 - 2	2 3 4 1 -1 -1 0 -2

Задача 5-3. На плоскости нарисовано n прямоугольников с вершинами в целых точках со сторонами, параллельными осям координат. Границы любых двух прямоугольников не имеют общих точек. То есть, другими словами, любые два прямоугольника либо расположены отдельно друг от друга, либо один из них — строго внутри другого. В такой ситуации, некоторые из прямоугольников — "внешние", т.е. такие что ни один из них не лежит внутри никакого другого прямоугольника, а остальные прямоугольники — "внутренние". Необходимо посчитать количество "внешних" прямоугольников.

В первой строке входа задано число n ($1 \le n \le 10^5$). В каждой из последующих n строк — по четыре целых числа x_1, y_1, x_2, y_2 ($-10^9 \le x_1, y_1, x_2, y_2 \le 10^9$), задающих координаты двух противоположных вершин соответствующего прямоугольника.

Выведите одно число — количество "внешних" прямоугольников.

Пример входа	Пример выхода
3 -3 -3 3 3 -2 2 2 -2 -1 -1 1 1	1
4 0 0 3 3 1 1 2 2 100 100 101 101 200 200 201 201	3

Задача 5-4. Рассмотрим случайную перестановку чисел от 1 до n: $\pi = (i_1, i_2, \ldots, i_n)$. Все перестановки при этом равновероятны. Построим *дучу* (*treap*) по парам $(1, i_1), (2, i_2), \ldots, (n, i_n)$ (здесь первая компонента пары обозначает ключ, а вторая — приоритет). Определите, с какой вероятностью высота дучи будет равна h. Высота дучи определяется как высота дерева. Высота дерева из одной вершины считается равной нулю.

В первой и единственной строке входа — два числа n и h ($1 \le n \le 100$, $0 \le h \le 10^9$). Ответ необходимо выдать с абсолютной или относительной погрешностью не более 10^{-6}. Чтобы избежать проблем с точностью, выводите ответ с десятью знаками после точки.

Пример входа	Пример выхода
1 0	1.0000000000
1 1	0.0000000000
3 2	0.6666666667