

## Задача А. Проверка сбалансированности (!) (1 балл)

Имя входного файла: `balance.in`  
Имя выходного файла: `balance.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

АВЛ-дерево является сбалансированным в следующем смысле: для любой вершины высота ее левого поддерева отличается от высоты ее правого поддерева не больше, чем на единицу.

Введем понятие *баланса вершины*: для вершины дерева  $V$  ее баланс  $B(V)$  равен разности высоты правого поддерева и высоты левого поддерева. Таким образом, свойство АВЛ-дерева, приведенное выше, можно сформулировать следующим образом: для любой ее вершины  $V$  выполняется следующее неравенство:

$$-1 \leq B(V) \leq 1.$$

Дано двоичное дерево поиска. Для каждой его вершины требуется определить ее баланс.

### Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число  $N$  ( $1 \leq N \leq 200000$ ) — число вершин в дереве. В последующих  $N$  строках файла находятся описания вершин дерева. В  $(i + 1)$ -ой строке файла ( $1 \leq i \leq N$ ) находится описание  $i$ -ой вершины, состоящее из трех чисел  $K_i$ ,  $L_i$ ,  $R_i$ , разделенных пробелами — ключа в  $i$ -ой вершине ( $|K_i| \leq 10^9$ ), номера левого ребенка  $i$ -ой вершины ( $i < L_i \leq N$  или  $L_i = 0$ , если левого ребенка нет) и номера правого ребенка  $i$ -ой вершины ( $i < R_i \leq N$  или  $R_i = 0$ , если правого ребенка нет).

Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

### Формат выходного файла

Для  $i$ -ой вершины в  $i$ -ой строке выведите одно число — баланс данной вершины.

### Пример

balance.in	balance.out
6	3
-2 0 2	-1
8 4 3	0
9 0 0	0
3 5 6	0
0 0 0	0
6 0 0	

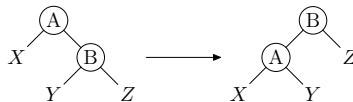
## Задача В. Делаю я левый поворот... (!) (2 балла)

Имя входного файла: `rotation.in`  
Имя выходного файла: `rotation.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

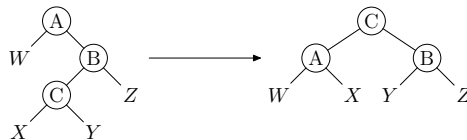
Для балансировки АВЛ-дерева при операциях вставки и удаления производятся *левые* и *правые* повороты. Левый поворот в вершине производится, когда баланс этой вершины больше 1, аналогично, правый поворот производится при балансе, меньшем  $-1$ .

Существует два разных левых (как, разумеется, и правых) поворота: *большой* и *малый* левый поворот.

Малый левый поворот осуществляется следующим образом:



Заметим, что если до выполнения малого левого поворота был нарушен баланс только корня дерева, то после его выполнения все вершины становятся сбалансированными, за исключением случая, когда у правого ребенка корня баланс до поворота равен  $-1$ . В этом случае вместо малого левого поворота выполняется большой левый поворот, который осуществляется так:



Дано дерево, в котором баланс корня равен 2. Сделайте левый поворот.

### Формат входного файла

Входной файл содержит описание двоичного дерева. В первой строке файла находится число  $N$  ( $3 \leq N \leq 200000$ ) — число вершин в дереве. В последующих  $N$  строках файла находятся описания вершин дерева. В  $(i+1)$ -ой строке файла ( $1 \leq i \leq N$ ) находится описание  $i$ -ой вершины, состоящее из трех чисел  $K_i, L_i, R_i$ , разделенных пробелами — ключа в  $i$ -ой вершине ( $|K_i| \leq 10^9$ ), номера левого ребенка  $i$ -ой вершины ( $i < L_i \leq N$  или  $L_i = 0$ , если левого ребенка нет) и номера правого ребенка  $i$ -ой вершины ( $i < R_i \leq N$  или  $R_i = 0$ , если правого ребенка нет).

Все ключи различны. Гарантируется, что данное дерево является деревом поиска. Баланс корня дерева (вершины с номером 1) равен 2, баланс всех остальных вершин находится в пределах от  $-1$  до 1.

### Формат выходного файла

Выведите в том же формате дерево после осуществления левого поворота. Нумерация вершин может быть произвольной при условии соблюдения формата. Так, номер вершины должен быть меньше номера ее детей.

### Пример

rotation.in	rotation.out
7	7
-2 7 2	3 2 3
8 4 3	-2 4 5
9 0 0	8 6 7
3 5 6	-7 0 0
0 0 0	0 0 0
6 0 0	6 0 0
-7 0 0	9 0 0

## Задача С. Вставка в АВЛ-дерево (2 балла)

Имя входного файла: addition.in  
Имя выходного файла: addition.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Вставка в АВЛ-дерево вершины  $V$  с ключом  $X$  при условии, что такой вершины в этом дереве нет, осуществляется следующим образом:

- находится вершина  $W$ , ребенком которой должна стать вершина  $V$ ;
- вершина  $V$  делается ребенком вершины  $W$ ;
- производится подъем от вершины  $W$  к корню, при этом, если какая-то из вершин несбалансирована, производится, в зависимости от значения баланса, левый или правый поворот.

Первый этап нуждается в пояснении. Спуск до будущего родителя вершины  $V$  осуществляется, начиная от корня, следующим образом:

- Пусть ключ текущей вершины равен  $Y$ .
- Если  $X < Y$  и у текущей вершины есть левый ребенок, переходим к левому ребенку.
- Если  $X < Y$  и у текущей вершины нет левого ребенка, то останавливаемся, текущая вершина будет родителем новой вершины.
- Если  $X > Y$  и у текущей вершины есть правый ребенок, переходим к правому ребенку.
- Если  $X > Y$  и у текущей вершины нет правого ребенка, то останавливаемся, текущая вершина будет родителем новой вершины.

Отдельно рассматривается следующий крайний случай — если до вставки дерево было пустым, то вставка новой вершины осуществляется проще: новая вершина становится корнем дерева.

### Формат входного файла

Входной файл содержит описание двоичного дерева, а также ключа вершины, которую требуется вставить в дерево.

В первой строке файла находится число  $N$  ( $0 \leq N \leq 200000$ ) — число вершин в дереве. В последующих  $N$  строках файла находятся описания вершин дерева. В  $(i+1)$ -ой строке файла ( $1 \leq i \leq N$ ) находится описание  $i$ -ой вершины, состоящее из трех чисел  $K_i$ ,  $L_i$ ,  $R_i$ , разделенных пробелами — ключа в  $i$ -ой вершине ( $|K_i| \leq 10^9$ ), номера левого ребенка  $i$ -ой вершины ( $i < L_i \leq N$  или  $L_i = 0$ , если левого ребенка нет) и номера правого ребенка  $i$ -ой вершины ( $i < R_i \leq N$  или  $R_i = 0$ , если правого ребенка нет).

Все ключи различны. Гарантируется, что данное дерево является корректным АВЛ-деревом.

В последней строке содержится число  $X$  ( $|X| \leq 10^9$ ) — ключ вершины, которую требуется вставить в дерево. Гарантируется, что такой вершины в дереве нет.

### Формат выходного файла

Выведите в том же формате дерево после осуществления операции вставки. Нумерация вершин может быть произвольной при условии соблюдения формата.

### Пример

addition.in	addition.out
2	3
3 0 2	4 2 3
4 0 0	3 0 0
5	5 0 0

## Задача D. Удаление из АВЛ-дерева (2 балла)

Имя входного файла: `deletion.in`  
Имя выходного файла: `deletion.out`  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Удаление из АВЛ-дерева вершины с ключом  $X$ , при условии ее наличия, осуществляется следующим образом:

- путем спуска от корня и проверки ключей находится  $V$  — удаляемая вершина;
- если вершина  $V$  — лист (то есть, у нее нет детей):
  - удаляем вершину;
  - поднимаемся к корню, начиная с бывшего родителя вершины  $V$ , при этом если встречается несбалансированная вершина, то производим поворот.
- если у вершины  $V$  не существует левого ребенка:
  - следовательно, баланс вершины равен единице и ее правый ребенок — лист;
  - заменяем вершину  $V$  ее правым ребенком;
  - поднимаемся к корню, производя, где необходимо, балансировку.
- иначе:
  - находим  $R$  — самую правую вершину в левом поддереве;
  - переносим ключ вершины  $R$  в вершину  $V$ ;
  - удаляем вершину  $R$  (у нее нет правого ребенка, поэтому она либо лист, либо имеет левого ребенка, являющегося листом);
  - поднимаемся к корню, начиная с бывшего родителя вершины  $R$ , производя балансировку.

Исключением является случай, когда производится удаление из дерева, состоящего из одной вершины — корня. Результатом удаления в этом случае будет пустое дерево.

Указанный алгоритм не является единственно возможным, но мы просим Вас реализовать именно его, так как тестирующая система проверяет точное равенство деревьев, получающихся в результате.

### Формат входного файла

Входной файл содержит описание двоичного дерева, а также ключа вершины, которую требуется удалить из дерева.

В первой строке файла находится число  $N$  ( $1 \leq N \leq 200000$ ) — число вершин в дереве. В последующих  $N$  строках файла находятся описания вершин дерева. В  $(i+1)$ -ой строке файла ( $1 \leq i \leq N$ ) находится описание  $i$ -ой вершины, состоящее из трех чисел  $K_i$ ,  $L_i$ ,  $R_i$ , разделенных пробелами — ключа в  $i$ -ой вершине ( $|K_i| \leq 10^9$ ), номера левого ребенка  $i$ -ой вершины ( $i < L_i \leq N$  или  $L_i = 0$ , если левого ребенка нет) и номера правого ребенка  $i$ -ой вершины ( $i < R_i \leq N$  или  $R_i = 0$ , если правого ребенка нет).

Все ключи различны. Гарантируется, что данное дерево является деревом поиска.

В последней строке содержится число  $X$  ( $|X| \leq 10^9$ ) — ключ вершины, которую требуется удалить из дерева. Гарантируется, что такая вершина в дереве существует.

### Формат выходного файла

Выведите в том же формате дерево после осуществления операции удаления. Нумерация вершин может быть произвольной при условии соблюдения формата.

### Пример

deletion.in	deletion.out
3	2
4 2 3	3 0 2
3 0 0	5 0 0
5 0 0	
4	

## Задача Е. Упорядоченное множество на АВЛ-дереве (3 балла)

Имя входного файла: avlset.in  
Имя выходного файла: avlset.out  
Ограничение по времени: 2 секунды  
Ограничение по памяти: 256 мегабайт

Если Вы сдали все предыдущие задачи, Вы уже можете написать эффективную реализацию упорядоченного множества на АВЛ-дереве. Сделайте это.

Для проверки того, что множество реализовано именно на АВЛ-дереве, мы просим Вас выводить баланс корня после каждой операции вставки и удаления.

Операции вставки и удаления требуется реализовать точно так же, как это было сделано в предыдущих двух задачах, потому что в ином случае баланс корня может отличаться от требуемого.

### Формат входного файла

В первой строке файла находится число  $N$  ( $1 \leq N \leq 200000$ ) — число операций над множеством. Изначально множество пусто. В каждой из последующих  $N$  строк файла находится описание операции.

Операции бывают следующих видов:

- А  $x$  — вставить число  $x$  в множество. Если число  $x$  там уже содержится, множество изменять не следует.
- D  $x$  — удалить число  $x$  из множества. Если числа  $x$  нет в множестве, множество изменять не следует.
- C  $x$  — проверить, есть ли число  $x$  в множестве.

### Формат выходного файла

Для каждой операции вида C  $x$  выведите Y, если число  $x$  содержится в множестве, и N, если не содержится.

Для каждой операции вида A  $x$  или D  $x$  выведите баланс корня дерева после выполнения операции. Если дерево пустое (в нем нет вершин), выведите 0.

Вывод для каждой операции должен содержаться на отдельной строке.

### Пример

avlset.in	avlset.out
6	0
A 3	1
A 4	0
A 5	Y
C 4	N
C 6	-1
D 5	