Задача B. Van Emde Boas tree Forever!

Имя входного файла:	`boas.in`
Имя выходного файла:	`boas.out`
Ограничение по времени:	8 секунд
Ограничение по памяти:	256 мегабайт

В этой задаче Вам необходимо эффективно реализовать интерфейс `TreeSet` для целых чисел в диапазоне $[0, 2^{31})$.

0. `insert` x — добавить в дерево ключ x
1. `delete` x — удалить из дерева ключ x (если он там есть!).
2. `find` x — если ключ x в дереве?
3. `next` x — минимальный элемент в дереве, строго больший x, или 30, если такого нет.
4. `prev` x — максимальный элемент в дереве, строго меньший x, или 30, если такого нет.

Формат входных данных

Из-за большого количества входных данных ввод в данной задаче осуществляется генераторами запросов.

В первой строке входного файла содержится натуральное число не более 10 — количество генераторов. В каждой следующей строке содержится описание генератора: 7 целых чисел n, a, b, m, c, d, e в диапазоне $[0, 2^{31}]$, а число $m \geqslant 1$.

Каждый генератор порождает n запросов к дереву, занумерованных числами от 1 до n. Положим $x_0 = 0, y_0 = 0,\ x_i = (ax_{i-1} + b_i) \mod m,\ y_i = (cy_{i-1} + d + ex_i) \mod 5$. Тогда i-ый запрос текущего генератора есть (x_i, y_i), где y_i — номер операции из списка выше, а x_i — её параметр. Следуйте формату входного файла из примера.

Гарантируется, что суммарное количество операций (сумма всех n по генераторам) не превосходит $7 \cdot 10^6$.

Формат выходных данных

В единственной строке выходного файла требуется вывести остатки от деления на 2^{32} следующих четырех чисел:

Первое — количество операций `find`, нашедших x в дереве.

Второе — сумма результатов операций `next`.

Третье — сумма результатов операций `prev`.

Четвертое — количество элементов в дереве к концу выполнения всех операций.

Напоминаем, что все числа нужно вывести по модулю 2^{32}.

Примеры

boas.in	boas.out
4 3 1 1 5 0 0 0 2 0 2 3 0 1 0 2 1 1 5 0 2 0 2 6210701 3 54321 2 3 0	1 30 1 2

Замечание

Во входном файле из примера задаётся следующая последовательность операций:

- `insert 1`
- `insert 2`
- `insert 3`
- `erase 2`
- `erase 2`
- `find 1`
- `find 2`
- `next 3`
- `prev 3`

После первых 5-ти операций множество, хранимое в дереве, будет выглядеть так: $\{1, 3\}$. Тогда результаты операций 6-9 будут: `true, false, 1, 30`, соответственно.