Задача 4-1. Во входе задано бинарное дерево поиска, такое что для любой вершины все ключи в ее левом поддереве строго меньше ее ключа, а все ключи в ее правом поддереве не меньше ее ключа. В первой строке — количество вершин в дереве, во второй — ключи дерева в порядке preorder. В выход выведите две строки. В первой — дерево в порядке postorder, во второй — дерево в порядке inorder. Количество вершин дерева не более 100 000. Значения ключей от 0 до 1 000 000 000.

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

Задача 4-2. В первой строке входа число n ($1 \le n \le 400$). Во второй строке — n целых чисел $a_1, a_2, \ldots, a_n$. Посчитайте количество различных бинарных деревьев поиска, состоящих из n вершин с ключами $a_1, a_2, \ldots, a_n$. Рассматриваются только такие деревья поиска, у которых в левом поддереве ключи строго меньше, а в правом — не меньше, чем в корне.

Когда два дерева считаются одинаковыми — интуитивно понятно: если нарисовать их на картинке, то картинки должны выглядеть одинаково. Формально это можно определить, например, так: два дерева считаются одинаковыми, если их preorder обходы порождают одну и ту же последовательность чисел. Ответ может быть очень большим числом, поэтому посчитайте его по модулю 123 456 789. Значения ключей от -2^{31} до $2^{31} - 1$.

Пример входа	Пример выхода
2 2 1	2
3 10 10 10	1
3 1 2 3	5

Задача 4-3.

Реализуйте следующий класс для хранения множества целых чисел:

```
class FixedSet {
 public:
  FixedSet();
  void Initialize(const vector<int>& numbers);
  bool Contains(int number) const;
};
```

`FixedSet` получает при вызове `Initialize` набор целых чисел, который впоследствии и будет хранить. Набор чисел не будет изменяться с течением времени (до следующего вызова `Initialize`). Операция `Contains` возвращает `true`, если число `number` содержится в наборе. Мат. ожидание времени работы `Initialize` должно составлять $O(n)$, где n —

количество чисел в `numbers`. Затраты памяти должны быть порядка $O(n)$. Операция `Contains` должна выполняться за $O(1)$.

С помощью этого класса решите модельную задачу: во входе будет дано множество различных чисел, а затем множество запросов — целых чисел. Необходимо для каждого запроса определить, лежит ли число из запроса в множестве.

В первой строке входа — число n — размер множества. $1 \le n \le 100\,000$. В следующей строке n различных целых чисел, по модулю не превосходящих 10^9. В следующей строке — число q — количество запросов. $1 \le q \le 1\,000\,000$. В следующей строке q целых чисел, по модулю не превосходящих 10^9. Для каждого запроса нужно вывести в отдельную строку “Yes” (без кавычек), если число из запроса есть в множестве и “No” (без кавычек), если числа из запроса нет в множестве. См. примеры.

Пример входа	Пример выхода
3 1 2 3 4 1 2 3 4	Yes Yes Yes No
3 3 1 2 4 10 1 5 2	No Yes No Yes

Задача 4-4. Вам дан набор из n треугольников ($1 \le n \le 1000000$). Каждый треугольник задается тремя целыми числами — длинами его сторон. Длины сторон — положительные целые числа, не превосходящие 10^9. Некоторые пары треугольников подобны. В силу свойств подобия известно, что все эти треугольники распадаются на несколько классов, в каждом из которых все треугольники подобны друг другу. К примеру, три треугольника со сторнами $(6, 6, 10)$, $(15, 25, 15)$, $(35, 21, 21)$ все попарно подобны друг другу, то есть образуют один класс. Необходимо найти количество классов подобия, на которые распадаются данные n треугольников. В первой строке входа дано число n. В каждой из следующих n строк — по три целых числа, задающих стороны очередного треугольника. Выведите число классов подобия. Сложность алгоритма должна в среднем линейно зависеть от n (может еще зависеть от ограничений на стороны треугольника). Т.е. никаких сортировок длинных массивов в решении делать нельзя.

Пример входа	Пример выхода
3 6 6 10 15 25 15 35 21 21	1
4 3 4 5 10 11 12 6 7 8 6 8 10	3