Задача 1 (Максимальное уникальное)

Считайте со стандартного потока ввода множество целых чисел.

Выведите максимальное из тех чисел, которые встречаются в этом множестве строго один раз.

Формат ввода

В первой строке целое число N - количество чисел.

Далее N строк, каждая содержит строго одно целое число, которое умещается в int.

Формат вывода

Одно целое число

Например:

Ввод	**Результат**
5 1 2 10 5 10	5

Задача 2 (СмЕшАнНыЙ регистр)

На стандартном потоке ввода приходят слова. Одинаковыми считаются слова, которые совпадают при сравнении без учёта регистра (например, "abc" и "AbC" - одинаковые). Одинаковые слова могут встречаться в потоке много раз в разном написании (например, всё те же "abc" и "AbC" по многу раз). Найти и вывести те слова, которые встречались в более чем двух вариантах написания.

Выводить в нижнем регистре. При выводе сортировать сперва по количеству вариантов написания от большего к меньшему, при равенстве количества вариантов - по возрастанию по словарю в нижнем регистре.

Формат ввода

В отдельной строке целое число N - количество слов. Далее N строк, в каждой одно слово.

Формат вывода

Найденные слова, отсортированные по условию. Каждое слово в новой строке.

Например:

Ввод	Результат
13 abc AbC foo BAR foo bar Foo bar bar bar ABC F00 f00	foo abc

Задача 3 (Экзамен)

Реализуйте класс, хранящий результаты экзамена и умеющий отвечать на запросы по ним. Прототип класса:

```
#include <string>
#include <set>

class ResultsHolder {
public:
    /**
     * Добавить в общую таблицу результат экзамена студента.
     * Параметры:
     * - full_name - строка с полным именем студента
     * - mark - оценка (от 1 до 10)
     * (Гарантируется, что совпадений полных имён у разных студентов не бывает.)
     *
     * Если студента с данным именем нет в результатах - добавить его.
     * Если студент с таким именем есть - обновить его результат
     * (потому что апелляции вполне возможны).
     */
    void update(const std::string& full_name, unsigned mark);

    /**
     * Вывести студентов, отсортировав по именам по алфавиту.
     *
     * Вывод в std::cout.
     *
     * Формат вывода:
     * Alex 7
     * Anastasia 9
     * Anny 5
     * Ivan 10
     * Nikita 8
     */
    void print_students() const;

    /**
     * Вывести студентов, отсортировав сперва по оценкам по убыванию,
     * а при равных оценках - по именам по алфавиту.
     *
     * Вывод в std::cout.
     *
     * Формат вывода:
     * Ivan 10
     * Anastasia 9
     * Nikita 8
     * Alex 7
     * Anny 5
     */
    void print_standings() const;

    /**
     * Обработать запрос военкомата.
     *
     * Военкомат передаёт список имён и хочет призвать этих людей в армию.
     * Если у этих людей неуд-ы (оценка ниже 3), то:
     * - вернуть эти имена в ответе военкомату;
     * - удалить этих людей из общей таблицы (потому что теперь они не студенты).
     *
     * Входной параметр: сет строк с именами, кого хочет призвать военкомат.
     * Возвращаемое значение: сет строк с именами, кого действительно можно призвать
     * (или пустой сет, если призвать никого нельзя).
     * Гарантируется, что военкомат не будет пытаться призвать несуществующих студентов.
```

```
     */
    std::set<std::string> process_military_request(const std::set<std::string>& names);
};
```

Для тестирования можете использовать следующий базовый пример:

```
    // Создали таблицу
    ResultsHolder rh;

    // Загрузили результаты
    rh.update("Alex", 6);
    rh.update("Anny", 5);
    rh.update("Ivan", 10);
    rh.update("Anastasia", 9);
    rh.update("Johnny", 1);
    rh.update("Alex", 7); // Апелляция у Alex, оценка обновляется
    rh.update("Nikita", 8);

    // Вывели в порядке убывания результата
    std::cout << "Exam results:" << std::endl;
    rh.print_standings();

    // Определили, кто идёт в армию
    auto to_military_service = rh.process_military_request({"Johnny", "Ivan"});

    // Вывели их на экран
    std::cout << "Are in army now:" << std::endl;
    for(const auto& s : to_military_service) {
        std::cout << s << std::endl;
    }

    // Вывели по алфавиту (но уже без Johnny, но Ivan остался с нами)
    std::cout << "Exam results:" << std::endl;
    rh.print_students();
```

Например:

Ввод	Результат
7 Alex 6 Anny 5 Ivan 10 Anastasia 9 Johnny 1 Alex 7 Nikita 8 2 Ivan Johnny	Exam results: Ivan 10 Anastasia 9 Nikita 8 Alex 7 Anny 5 Johnny 1 MoD request: Ivan Johnny Are in army now: Johnny Next year group: Alex 7 Anastasia 9 Anny 5 Ivan 10 Nikita 8

Задача 4 (Один сломал, второй потерял)

История

Поймали канибалы русского, американца и англичанина и говорят: "Двоих из вас мы съедим, а одного оставим в живых - того, кто больше всех нас удивит". Посадили их в отдельные пустые комнаты и дали каждому по два 20 киллограмовых металлических шара. Сказали, что при помощи этих шаров они и должны к завтрашнему дню всех удивить.

Проходит ночь. Приходят вождь со свитой к американцу, а тот этими шарами жонглирует по всякому. Круто. Пошли к англичанину, а он на ногах на этих шарах катается. Класс. Пошли к русскому. Смотрят - он сидит в пустой комнате. Где, спрашивают, шары? А он в ответ: "Один сломал, другой потерял".

Условия задачи

Реализуйте класс Ball, описывающий металлический шар. В ходе работы программы таких шаров может быть создано сколь угодно много.

Любой шар может быть в одном из трёх состояний: целый, сломанный, потерянный. Исходно шары создаются в целом состоянии.

Из всех созданных шаров строго один шар должно быть можно сломать, и строго один - потерять. Ломать и терять можно только целые шары.

Попытка сломать или потерять второй шар должна приводить к исключению. Попытка сломать или потерять шар, который уже сломанный или потерянный, тоже должна приводить к исключению.

Прототип класса:

```
class Ball {
public:
    // Конструктор и деструктор шара, если нужны

    // Попытка сломать шар.
    // Для первого целого шара должна заканчиваться успешно.
    // Если один шар уже был сломан, выбросить std::exception
    // Если пытаемся ломать уже сломанный или потерянный шар, выбросить std::exception
    void destroy();

    // Попытка потерять шар.
    // Для первого целого шара должна заканчиваться успешно.
    // Если один шар уже был потерян, то выбросить std::exception
    // Если пытаемся терять уже сломанный или потерянный шар, выбросить std::exception
    void lose();
};
```

Например:

Ввод	Результат
2 1 1 0 1	Break attempt #0, trying to break ball 0: success! Lose attempt #0, trying to lose ball 1: success!

Задача 5 (История объекта)

На стандартном потоке ввода приходит поток событий, происходивших с полями некоторого объекта. Каждое событие имеет следующий формат:

```
timestamp action property value
```

Здесь:

- timestamp - метка времени, целое число без знака, отсчитывается от начала времён;
- action - действие над полем, может быть SET (задание значения поля) или DELETE (удаление поля);
- property - имя поля, произвольная строка;
- value - значение поля, для событий SET является произвольной строкой и задаёт новое значение поля, для событий DELETE пустое.

Гарантируется, что события упорядочены по неубыванию timestamp. Совпадения timestamp событий возможны, но при этом совпадения могут быть только для событий с разными property.

Пример потока событий:

```
100 SET status starting
100 SET pressure 10
104 SET status on
105 SET speed 42
110 SET pressure 20
111 SET status overload
119 SET status breaking
120 DELETE status
149 SET status repaired
150 SET status ready
```

После конца потока событий приходит поток запросов - показать полное состояние полей объекта в заданный момент времени. Формат каждого запроса - timestamp, для которого нужно показать ответ.

Изменение, пришедшее в момент N, становится видно в момент N+1. То есть, если по логу свойство появилось в момент 100, то в момент 100 его ещё не видно, а в момент 101 - уже видно.

Например, для приведённой выше истории изменений в момент 100 объект пустой, в момент 103 объект имеет поля `pressure=10` и `status=starting`, а в момент 125 поля `pressure=20` и `speed=42`.

Требуется обработать историю событий и по ней ответить на запросы. При ответе на запрос печатать поля объекта в формате "имя_поля : значение", через запятую и пробел, поля сортировать по алфавиту по именам (см. пример ниже).

Формат ввода

Сначала в отдельной строке целое число N - количество событий. Затем N строк с событиями описанной структуры. Далее в отдельной строке M - количество запросов. Затем M строк с запросами, в каждой одно целое число.

Формат вывода

M строк с ответами на запросы в описанном формате.

Например:

Ввод	Результат
10 100 SET status starting 100 SET pressure 10 104 SET status on 105 SET speed 42 110 SET pressure 20 111 SET status overload 119 SET status breaking 120 DELETE status 149 SET status repaired 150 SET status ready 6 90 100 103 111 125 180	pressure : 10, status : starting pressure : 20, speed : 42, status : on pressure : 20, speed : 42 pressure : 20, speed : 42, status : ready