Отборочный этап. Второй тур (приведен один из вариантов заданий)

1. Электронные таблицы. Адресация ячеек и вычисления (1 балл) [3x3]

Дан фрагмент электронной таблицы:

	A	B	C	D	E
1	1	2	5	11	
2	3				
3	7				
4	13				
5					

В ячейку B2 поместили формулу вида =#A#1*#B#1*#A#2, где на месте каждого знака # может быть или не быть символ $.

Ячейку B2 скопировали во все ячейки диапазона B2:D4.

Определите конкретную формулу, которая была в ячейке B2, если известно, что в ячейке D4 получилось значение 11466. В ответе укажите последовательность из шести двоичных разрядов, означающих наличие (1) или отсутствие (0) символов $ на позициях, отмеченных знаком # считая слева направо. Например, ответ 111001 соответствует формуле =A1*$B1*A$2.

Если такой формулы не существует, укажите в ответе NULL.

Ответ: 101010

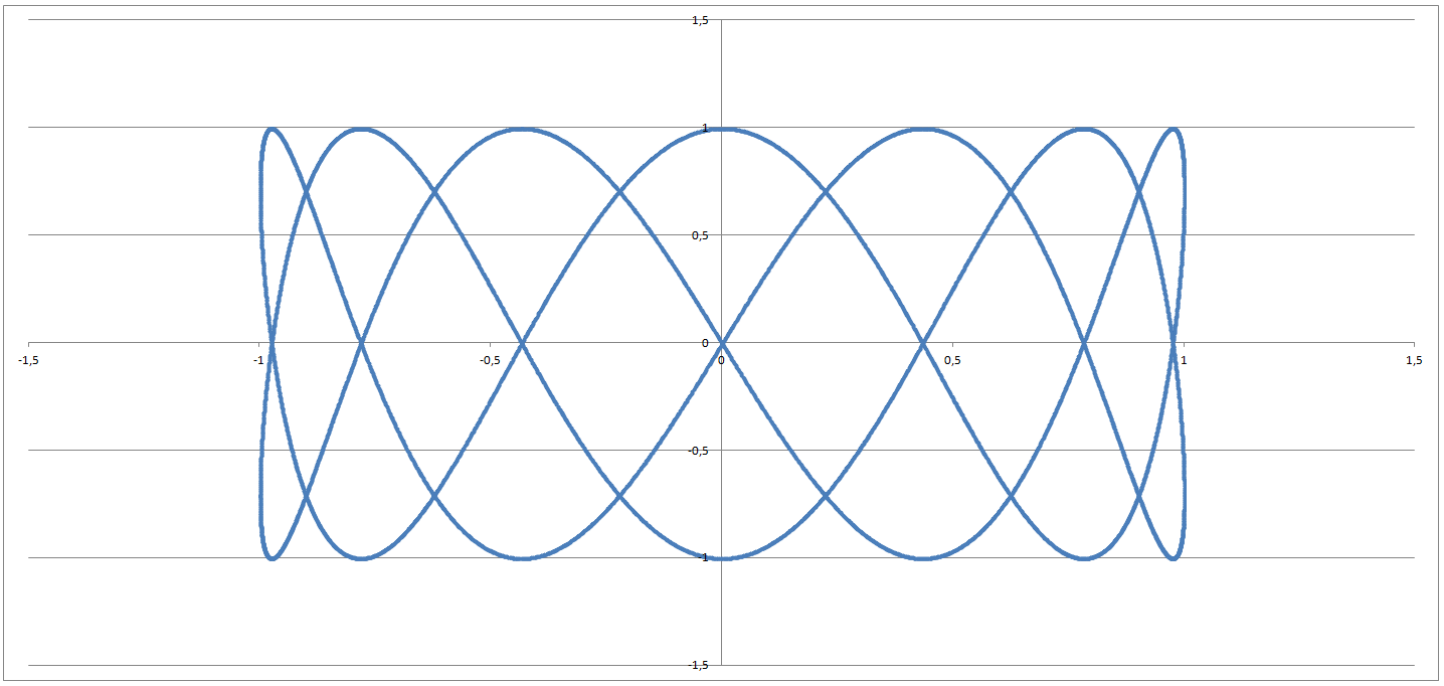
2. Электронные таблицы. Графики и диаграммы (2 балла) [Фора физикам]

Дан фрагмент электронной таблицы в режиме отображения формул:

	A	B	C	D
1		[redacted]		
2	0	=SIN(B$1*A2)	=SIN(C$1*A2)	
3	0,001			
4	0,002			
5	0,003			
6	0,004			
7	0,005			
8	0,006			

Ячейки диапазона A2:A10000 заполнены числами, образующими арифметическую прогрессию с шагом 0,001.

Ячейку B2 скопировали во все ячейки диапазона B3:B10000. Ячейку C2 скопировали во все ячейки диапазона C3:C10000. Затем выделили диапазон B2:C10000 и построили по нему точечную диаграмму. Получился следующий результат:



Определите числа, которые находятся в ячейках B1 и C1, если про них известно следующее:

1. Оба числа являются натуральными.
2. Хотя бы одно число больше 1000.
3. Сумма этих чисел минимально возможная.

В ответе укажите через пробел сначала значение в ячейке B1 и затем значение в ячейке C1.

Ответ: 286 1001

3. Сортировка и фильтрация данных (2 балла)

[Продажи]

Таблица Orders в реляционной базе данных содержит записи о заказах, состоящие из значений трех атрибутов:

1. Customer – имя заказчика, может принимать значения только A, B или C.
2. Product – название продукта, может принимать только значения P1, P2 или P3.
3. Quantity – количество единовременно заказанного продукта конкретным заказчиком, может принимать любое целое положительное значение. Каждый заказчик может неограниченное количество раз заказывать некоторое количество любого продукта, о чем добавляется очередная запись в таблицу.

Известны результаты выполнения нескольких SQL запросов:

SELECT SUM(Quantity) FROM Orders GROUP BY Customer ORDER BY Customer	70 65 45
SELECT SUM(Quantity) FROM Orders GROUP BY Product ORDER BY Product	65 55 60
SELECT SUM(Quantity) FROM Orders WHERE Customer IN (A, B) GROUP BY Product ORDER BY Product	25 50 60
SELECT SUM(Quantity) FROM Orders WHERE Product=P1 and Customer=A	15
SELECT SUM(Quantity) FROM Orders WHERE Product=P3 and Customer=B	25

Что будет выведено в результате выполнения запроса:
SELECT SUM(Quantity) FROM Orders WHERE Product=P2 and Customer=A

Примечание. Семантика использованных операторов:

SELECT ###	Выводит значения атрибутов или функций, вычисленных от множества значений атрибутов, перечисленных после ключевого слова SELECT
FROM ###	Указывает название таблицы, из которой берутся значения
WHERE ###	Фильтр. В вывод SELECT, в том числе в качестве аргументов вычислимых функций, будут попадать только записи, удовлетворяющие условиям фильтра. Конструкция X IN (T1, T2, …, TN) означает, что фильтру удовлетворяют

	только записи, в которых атрибут X равен одному из перечисленных в скобках значений.
GROUP BY ###	Группирует записи так, что в одной группе оказываются все записи с одинаковыми значениями атрибута ###. В этом случае функция, указанная в SELECT вычисляется отдельно для каждой группы.
SUM(###)	Функция, вычисляющая сумму значений атрибута ### для всех записей с учетом фильтра и группировки, если они используются.
ORDER BY ###	Сортирует выводимые результаты по возрастанию (для атрибутов, являющихся строками – в лексикографическом порядке), в том числе по атрибуту, который сам не выводится оператором SELECT.

Ответ: 20

4. Мультимедиа технологии (3 балла)

[HSB]

В компьютерной графике для растрового изображения часто строят гистограмму яркостей по каналам. Она представляется собой три графика для красного, зеленого и синего каналов соответственно, где на каждом графике по оси абсцисс отложены возможные значения яркости пикселей в канале изображения (слева направо от 0 до 255), а по оси ординат количество точек, имеющих такую яркость в выбранном канале.

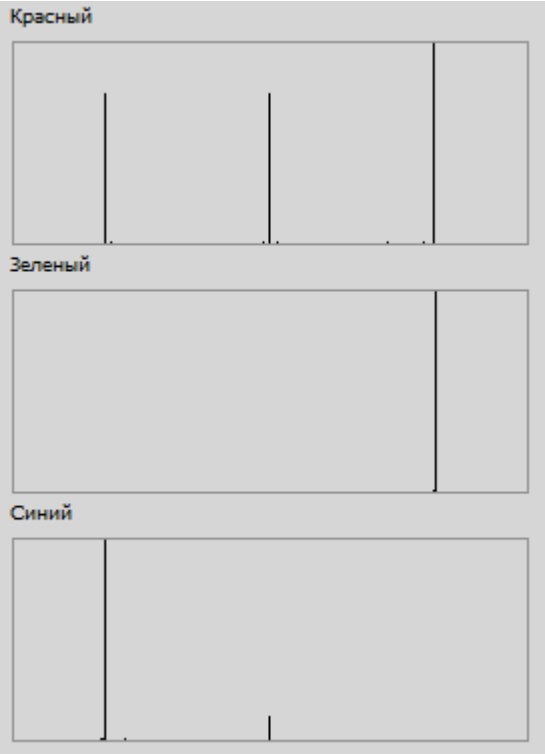
Известно, что исходное изображение состоит из 20 равных по количеству пикселей областей, окрашенных следующим образом:

Цвет	Количество областей на изображении
Красный=250, Зеленый=5, Синий=5	8
Красный=250, Зеленый=128, Синий=5	6
Красный=250, Зеленый=250, Синий=5	4
Красный=128, Зеленый=250, Синий=5	2

Распространенным инструментом цветокоррекции является изменение значение цветов пикселей в цветовой модели HSB. В этом случае для пикселей изображения могут применяться операции изменения цветового тона (Hue) от –180° до +180°, изменения насыщенности (Saturation) от –100 до +100 (в процентах), изменения яркости (Brightness) от –100 до +100 (в процентах). Пусть задан набор операций цветокоррекции, которые можно применять к изображению:

Номер операции	Операция
1	Hue +60°
2	Hue +120°
3	Hue +180°
4	Hue –60°
5	Hue –120°
6	Saturation –33
7	Saturation –67

Известно, что к исходному изображению последовательно применили ровно две операции, после чего получили следующие гистограммы по каналам:



Определите примененные операции и укажите их номера в ответе через пробел в порядке возрастания номеров.

Ответ: 1 6

5. Телекоммуникационные технологии (2 балла). [DDoS-атака]

В вычислительном центре университета стоит высокопроизводительный сервер, обрабатывающий запросы двух типов. Все принятые запросы становятся в единую бесконечно большую очередь на обработку. На определение типа запроса у сервера уходит 3 миллисекунды, если запрос оказался известного типа, он сразу же отправляется на обработку, в противном случае он игнорируется. Запрос типа 1 обрабатывается 7 миллисекунд, а запрос типа 2 обрабатывается 12 миллисекунд. Определение типа запроса и его обработка выполняются последовательно для каждого запроса. Это значит, что если в очереди сервера есть два запроса, то сначала будет определён тип первого запроса, затем при необходимости первый запрос будет обработан, и только потом будет определяться тип второго запроса.

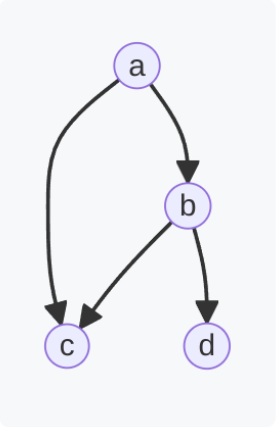
Злоумышленник пытается вывести из строя сервер, посылая случайные запросы каждую миллисекунду. При этом гарантируется, что случайный запрос не может совпасть с запросами типа 1 и типа 2 из-за их сложной структуры. В то же время пользователь сервера отправил 9 запросов обоих типов с интервалом в 15 миллисекунд в следующем порядке: {1 1 2 1 1 2 1 1 2}. И злоумышленник, и пользователь начали отправлять свои запросы одновременно в момент времени $t_0 = 0$.

Для защиты сервера администратор подключил защиту от DDoS-атак. Защита работает следующим образом: все принятые запросы становятся в единую бесконечно большую очередь на обработку. Каждый входящий запрос берётся из очереди и анализируется в течение некоторого времени и либо блокируется, либо пропускается дальше к вычислительному серверу. Обработка первых 30 запросов осуществляется за 2 миллисекунды на каждый запрос для определения факта атаки, и все эти запросы будут пропущены в очередь сервера независимо от типа. Обработка каждого следующего запроса осуществляется за 1 миллисекунду, при этом тип запроса не важен, но запросы от злоумышленника не пропускаются. При этом защита от DDoS-атак срабатывает не всегда и все-таки пропускает в очередь сервера каждый пятидесятый запрос злоумышленника, при этом 49 предыдущих его запросов блокируются. Защита от DDoS-атак анализирует запросы параллельно с работой сервера.

Определите, через сколько миллисекунд будет обработан последний запрос пользователя. Временем на передачу данных между узлами сети следует пренебречь (то есть если кто-то отправил запрос, он тут же в начале миллисекунды будет принят сервером, поставлен в очередь и при необходимости будет начата его обработка). Если на анализ поступает несколько пакетов в течение одной миллисекунды, в очередь сначала попадает пакет злоумышленника.

Ответ: 194

6. Операционные системы (3 балла) [Сколько вам пакетов?]



В операционных системах семейства GNU/Linux для установки ПО используется утилита, которая называется **менеджером пакетов** (или пакетным менеджером). Основная задача менеджера пакетов - определить, от каких пакетов зависит устанавливаемый пакет, и установить их вместе с ним. Такая операция производится рекурсивно для каждой зависимости, и в конце концов менеджер пакетов получает полный список зависимостей, который можно представить в виде графа. Такая операция называется **разрешением зависимостей**. Например, если для

установки пакета a нужны пакеты b и c, а для пакета b нужно установить пакеты c и d, то в итоге получится граф, изображённый на рисунке справа.

В данный момент сообществом разрабатывается экспериментальный пакетный менеджер. Рассмотрим принцип его работы.

Для работы менеджеру нужно иметь информацию о доступных пакетах в виде текстового файла определённого формата. Первая строчка описания каждого пакета начинается с ключевого слова `Package:`, после которого указывается название пакета. Вторая строчка описания содержит количество доступных версий и начинается с ключевого слова `Available-Versions:`. После этой строчки указывается информация о версиях. Каждая версия описывается тремя параметрами: номер версии (начинается с ключевого слова `Version:`), список необходимых зависимостей (начинается с ключевого слова `Depends:`) и размер скачиваемого файла (начинается с ключевого слова `Size:`). Пример информации о пакете с двумя версиями без зависимостей:

```
   Package: gcc-12-lib
Available-Versions: 2
Version: 12.3.0
Depends:
Size: 20068
Version: 12.2.8
Depends:
Size: 18880
```

При наличии зависимостей они описываются с использованием определённого формата. После ключевого слова `Depends:` через запятую указывается список пакетов, от которых зависит описываемый. Для каждой зависимости указывается её название, одна из операций сравнения и номер версии. Зависимости перечисляются через запятую. Пример информации о пакете с одной версией и двумя зависимостями:

```
   Package: libc6
Available-Versions: 1
Version: 2.35.5
Depends: libgcc-s1 >= 12.0.0, libcrypt1 >= 4.4.10
Size: 20068
```

Если в файле описано несколько пакетов, тогда описания разделяются пустой строкой.

Номер версии всегда состоит из трёх чисел, разделённых точкой. Первое число называется мажорной версией, второе - минорной версией, третье - патч-версией. Например, если версия равна 2.3.7, то мажорная версия равна 2, минорная - 3, патч-версия - 7.

Пакетный менеджер предусматривает следующие операции сравнения версий:

Операция	Описание
>>	Строго больше
>=	Больше или равна
==	Строго равна
~=	Больше или равна с сохранением мажорной и минорной версий
^=	Больше или равна с сохранением мажорной версии

При сравнении версий сначала сравниваются мажорные версии, затем минорные, затем патч-версии.

Граф зависимостей должен удовлетворять следующим правилам:

Зависимости каждого из пакетов в графе также должны быть в этом графе.

Изначально при установке пакета выбирается максимально возможная версия, подходящая под ограничения.

Если какая-то зависимость объявляется несколько раз (например, в разных пакетах), то версия пакета в итоговом варианте графа должна удовлетворять всем ограничениям.

Пакет нельзя установить, если какой-то из его зависимостей не существует либо не существует подходящей под ограничения версии зависимости. В таком случае нужно понижать версию проблемной зависимости и пробовать строить граф до тех пор, пока это не получится сделать. Если понижение версии зависимости не помогло, нужно понижать версию самого пакета, в случае необходимости продвигаясь по дереву вверх и выполняя понижение версии для всех пакетов на пути поочерёдно.

Необходимо построить граф зависимостей, соответствующий сформулированным требования, для пакета `zsh`.

Вам даётся доступ к поисковой строке, которая по названию пакета находит его служебную информацию в формате, описанном выше. Также Вы можете скачать текстовый файл https://olymp.itmo.ru/files/2024-01/5a59/4bcb/350bec6f-9d79-bf1d3a552c5a.txt с описанием всех пакетов. Определите, сколько пакетов будет установлено и каков общий размер скачиваемых файлов. В ответе введите два числа через пробел.

Ответ: 8 8309339

7. Технологии программирования (2 балла) [MindCraft]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

В новой популярной игре «MindCraft» игрок управляет множеством фабрик по производству ресурсов. А конечная цель игры — производить определенные артефакты, для каждого из которых нужно фиксированное количество единиц определенных ресурсов. Всего есть n ресурсов, и i-я фабрика производит только i-й ресурс. Игрок может запускать и останавливать любые фабрики в любой момент времени. **Изначально все фабрики запущены.**

Также в игре есть особенный ресурс — *энергия*. Энергия нужна, чтобы фабрики работали: если энергии не хватает, фабрика не произведет соответствующий ресурс. Изначально у игрока есть A единиц энергии.

Каждый ход, именно в таком порядке:

игрок получает E единиц энергии;

игрок может запустить или остановить какие-то фабрики (можно ничего не менять);

запущенные фабрики последовательно выполняют работу: сначала первая, затем вторая, и т.д., и в самом конце — n-я;

выполняя работу, каждая запущенная в данный момент фабрика тратит энергию и производит ресурс: i-я фабрика тратит за ход ровно e_i энергии и производит x_i единиц i-го ресурса;

если какой-то фабрике не хватило энергии (текущее количество энергии у игрока строго меньше e_i), она в этот ход не работает.

В конце хода игрок может попробовать произвести какие-то артефакты. Всего есть m возможных артефактов, и для каждого из них известен его *рецепт* — какие ресурсы и в каком количестве нужны для его производства. При успешном производстве артефакта количество единиц каждого ресурса уменьшается на необходимую для производства величину. Если какого-то ресурса не хватает для производства, оно считается проваленным, и ресурсы не расходуются.

Просимулируйте действия игрока, и на каждый запрос производства артефакта сообщите, успешно ли он выполнен, и если нет — каких ресурсов не хватает.

Формат входных данных

Решите задачу для t уровней игры. В первой строке ввода дано целое число t — количество уровней, для которых требуется решить задачу ($1 \le t \le 40$). Далее следуют t описаний уровней.

Первая строка описания уровня содержит три целых числа A, E и n — изначальное количество энергии, сколько энергии добавляется каждый ход, и сколько типов ресурсов (и фабрик) есть в игре ($0 \le A, E \le 10^5$; $1 \le n \le 50$). В i-й из следующих n строк через пробел даны два целых числа e_i и x_i — расход энергии и количество единиц производимого за ход ресурса для i-й фабрики ($1 \le e_i, x_i \le 10^4$).

В следующей строке дано целое число m — количество артефактов в игре ($1 \le m \le 50$). В i-й из следующих строк дан рецепт i-го артефакта — n целых чисел $r_{i,j}$, записанных через пробел, означающих количество ресурса номер j, необходимое для производства этого артефакта ($0 \le r_{i,j} \le 10^5$).

В следующей строке дано целое число q — количество действий игрока ($1 \le q \le 1000$). В следующих q строках даны описания действий игрока. Действия даны в хронологическом порядке, i-е действие задается в формате «s_i ON f_i» или «s_i OFF f_i», если описывает запуск или остановку фабрики с номером f_i на ходу номер s_i, и в формате «s_i CREATE a_i», если описывает запрос на производство артефакта с номером a_i в конце хода s_i ($1 \le s_i \le 1000$; $1 \le f_i \le n$; $1 \le a_i \le m$).

Формат выходных данных

Для каждого запроса на производство артефакта выведите в отдельной строке «OK» (без кавычек), если артефакт был успешно произведен, иначе — выведите «FAIL: missing a», где a — **минимальный** номер ресурса, количества которого не хватает для его производства.

Пример

Стандартный ввод	Стандартный вывод
3 10 6 3 2 3 3 4 1 1 5 2 4 2 1 1 1 2 2 2 1 1 1 1 1 1 5 2 CREATE 1 3 CREATE 1 4 OFF 2 5 OFF 1 6 CREATE 5 10 3 4 1 6 3 1 8 2	OK FAIL: missing 3 OK FAIL: missing 4 FAIL: missing 3 OK FAIL: missing 2 OK

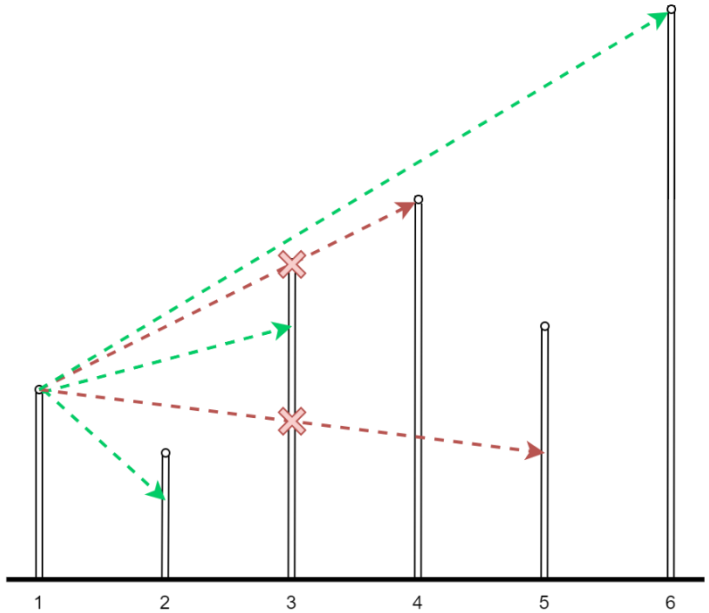
Стандартный ввод	Стандартный вывод
3 2 3 2 2 2 1 2 1 3 2 2 0 1 1 6 2 CREATE 1 3 OFF 1 3 OFF 3 8 CREATE 2 9 OFF 2 12 CREATE 3 1 15 3 4 6 8 2 6 3 2 3 2 1 3 7 4 4 3 CREATE 2 4 OFF 1 11 ON 1 12 CREATE 2	

8. Технологии программирования (4 балла) [Видеоконференция]

Имя входного файла	стандартный ввод
Имя выходного файла	стандартный вывод
Ограничение по времени	2 секунды
Ограничение по памяти	256 мегабайт

В Байтландии на одной из улиц в ряд стоят n небоскребов. Небоскреб номер i расположен в x_i метрах от начала улицы и имеет высоту y_i метров. Для простоты будем представлять i-й небоскреб как вертикальный отрезок длины y_i с нижним из двух концов на уровне земли.

На крыше первого небоскреба планируют разместить видовую площадку. Чтобы площадка действительно была видовой, с нее должно быть видно как можно больше других небоскребов на главной улице. Мы будем считать, что небоскреб i *виден* из точки (x_1, y_1), если существует отрезок с одним из концов в точке (x_1, y_1) и вторым концом на i-м небоскребе, не пересекающий ни один другой небоскреб (см. рисунок).



С крыши первого небоскреба видны: второй, третий и шестой.

Правительство города готово снести произвольное количество небоскребов, чтобы обеспечить наилучший вид с видовой площадки. Определите, какое максимальное количество небоскребов может быть видно с крыши первого, если разрешается снести произвольное их количество.

Формат входных данных

Решите задачу для t различных улиц Байтландии. В первой строке ввода дано целое число t — количество улиц, для которых надо решить задачу ($1 \le t \le 100$). Далее следуют t описаний улиц.

В первой строке описания улицы дано целое число n — количество небоскребов ($2 \le n \le 1000$). В i-й из следующих n строк через пробел даны целые числа x_i и y_i — расстояние от начала улицы до i-го небоскреба и его высота, соответственно ($0 \le x_i \le 10^4$; $1 \le y_i \le 10^4$). Гарантируется, что все x_i различны и что небоскребы перечислены в порядке возрастания x_i.

Гарантируется, что сумма n по всем t улицам не превосходит 1000.

Формат выходных данных

Выведите по очереди ответы на задачу для каждой из t улиц. В первой строке ответа выведите целое число k — максимальное количество небоскребов, которые может быть видно с видовой площадки. Во второй строке перечислите через пробел k различных целых чисел от 2 до n — номера этих небоскребов.

Обратите внимание, что требуется вывести именно номера тех небоскребов, которые будут видны с площадки, а не номера тех небоскребов, которые следует снести.

Пример

Стандартный ввод	Стандартный вывод
3 4 1 1 2 3 3 2 4 4 4 1 1 2 4 3 9 4 16 6 1 1 2 2 3 5 4 2 10 5 15 8	2 3 4 3 2 3 4 3 4 5 6