

## Общая информация по задачам второго тура

### Доступ к результатам проверки решений задач во время тура

В течение тура можно не более 10 раз по каждой задаче запросить информацию о результатах оценивания решения на тестах жюри. Запрос по каждой задаче можно делать не чаще одного раза в 5 минут.

### Ограничение на размер исходного кода программы-решения

Во всех задачах размер файла с исходным кодом решения не должен превышать 256 КБ.

### Сводная таблица ограничений

Задача	Ограничение по времени	Ограничение по памяти	Получение результатов во время тура
5. Звёздный путь	1 секунда	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
6. Морской бой	2 секунды	256 МБ	Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте
7. Массовый прогноз	1 секунда	256 МБ	Результаты окончательной проверки не сообщаются
8. Флешмоб	2 секунды	256 МБ	Для каждой подзадачи сообщаются только баллы за эту подзадачу

## Задача 5. Звёздный путь

Имя входного файла:	expedition.in
Имя выходного файла:	expedition.out
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 МБ

Экспедиция готовится отправиться в путь на космическом корабле нового поколения. Планируется последовательно посетить  $N$  планет звёздной системы — от планеты Земля до планеты Победа. Планеты пронумерованы от 1 до  $N$  в порядке их посещения, Земля имеет номер 1, а Победа — номер  $N$ .

Для перелёта между планетами корабль может использовать любой тип топлива, существующий в звёздной системе. Перед началом экспедиции корабль находится на планете Земля, и бак корабля пуст. Существующие типы топлива пронумерованы целыми числами, на планете с номером  $i$  можно заправиться только топливом типа  $a_i$ . При посещении  $i$ -й планеты можно заправиться, полностью освободив бак от имеющегося топлива и заполнив его топливом типа  $a_i$ .

На каждой планете станция заправки устроена таким образом, что в бак заправляется ровно столько топлива, сколько потребуется для перелёта до следующей планеты с топливом такого же типа. Если далее такой тип топлива не встречается, заправиться на этой планете невозможно. Иначе говоря, после заправки на  $i$ -й планете топлива хватит для посещения планет от  $(i + 1)$ -й до  $j$ -й включительно, где  $j$  — минимальный номер планеты, такой что  $j > i$  и  $a_j = a_i$ . Для продолжения экспедиции дальше  $j$ -й планеты корабль необходимо снова заправить на одной из этих планет.

Требуется написать программу, которая по заданным типам топлива на планетах определяет минимальное количество заправок, требуемых для экспедиции.

### Формат входных данных

В первой строке входного файла записано число  $N$  ( $2 \leq N \leq 300\,000$ ) — количество планет.

Во второй строке входного файла записано  $N$  целых чисел  $a_1, a_2, \dots, a_N$  ( $1 \leq a_i \leq 300\,000$ ) — типы топлива на планетах.

### Формат выходных данных

В первой строке выходного файла выведите единственное число  $K$  — минимальное количество заправок, которые нужно произвести.

Во второй строке выведите  $K$  чисел, разделённых пробелами, — номера планет, на которых требуется заправиться. Номера планет требуется выводить в порядке времени заправок.

Если решений с минимальным количеством заправок несколько, выведите любое из них. Если решения не существует, выведите число 0.

### Система оценивания

Данная задача содержит две подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы успешно пройдены.

#### Подзадача 1

$N \leq 3000$ . Подзадача оценивается в 50 баллов.

#### Подзадача 2

$N \leq 300\,000$ . Подзадача оценивается в 50 баллов.

### Примеры

expedition.in	expedition.out
7 1 3 2 1 3 2 3	3 1 3 5
7 4 3 2 4 3 2 1	0

## Задача 6. Морской бой

Имя входного файла:	<i>стандартный ввод</i>
Имя выходного файла:	<i>стандартный вывод</i>
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 МБ

В рамках Чемпионата Урала планируется проведение турнира стратегий по игре «Морской бой 1D».

Игра проходит на поле, которое представляет собой прямоугольник размером  $1 \times N$  клеток. На поле расставляются  $T$  кораблей, каждый из которых имеет вид прямоугольника размером  $1 \times K$  клеток. Расстановка кораблей на поле является *допустимой*, если различные корабли не имеют общих клеток и разделены хотя бы одной пустой клеткой. Игровая программа осуществляет выстрелы в клетки поля, а сервер сообщает, является ли выстрел промахом или попаданием в корабль.

В процессе игры про некоторые клетки становится известно, что при любой допустимой расстановке кораблей они принадлежат какому-либо из кораблей. Назовём такие клетки *заведомо занятыми*.

Игра заканчивается после первого попадания в корабль. Сервер пытается добиться того, чтобы игра продолжалась как можно дольше. Для этого он не фиксирует расстановку кораблей в начале игры, а рассматривает все возможные допустимые расстановки и сообщает о попадании, только если клетка, в которую осуществляется выстрел, является заведомо занятой.

Требуется написать программу, исполняющую роль сервера для этой игры. Сервер сначала загружает параметры игры, а затем взаимодействует с игровой программой, сообщая после каждого выстрела информацию о промахе или попадании, а также количество заведомо занятых клеток.

### Протокол взаимодействия

**Задача является интерактивной.** После каждого вывода требуется сбросить буфер вывода.

Роль игровой программы исполняет программа жюри. Программа-решение исполняет роль сервера.

Первая строка стандартного ввода программы-решения содержит параметры игры — три числа:  $N$  — размер игрового поля,  $T$  — число кораблей и  $K$  — длина каждого корабля ( $1 \leq N \leq 100\,000$ ,  $1 \leq T$ ,  $1 \leq K$ ). Гарантируется, что на поле длины  $N$  можно по описанным правилам разместить  $T$  кораблей длины  $K$ .

После считывания параметров игры программа-решение должна определить и вывести в стандартный поток вывода количество заведомо занятых клеток.

Затем начинается игра. Программа-решение должна последовательно считывать ходы игровой программы из стандартного потока ввода и обрабатывать их следующим образом:

1. Считать из стандартного потока ввода одно число  $q$  — номер клетки, в которую игровая программа осуществляет выстрел ( $1 \leq q \leq N$ ). Игровая программа никогда не делает два выстрела в одну и ту же клетку.
2. Если клетка  $q$  является заведомо занятой, вывести в стандартный поток вывода число 1 и завершить работу.
3. Если клетка  $q$  не является заведомо занятой, вывести в стандартный поток два числа, разделенных пробелом: 0 и количество заведомо занятых клеток после этого выстрела.

После этого программа-решение переходит к пункту 1 и продолжает взаимодействие с игровой программой.

### Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы успешно пройдены.

#### Подзадача 1

$N \leq 15$ . Подзадача оценивается в 30 баллов.

#### Подзадача 2

$N \leq 3000$ . Подзадача оценивается в 30 баллов.

#### Подзадача 3

$N \leq 100\,000$ . Подзадача оценивается в 40 баллов.

## Пример взаимодействия

стандартный ввод	стандартный вывод
8 2 3	4
4	0 5
1	1

## Пояснение к примеру

Игра происходит на поле из 8 клеток, на котором расставляются 2 корабля, состоящие из 3-х клеток каждый. Все допустимые расстановки кораблей приведены на рис. 1. Клетки, отмеченные «#», заведомо заняты. Таких клеток 4.

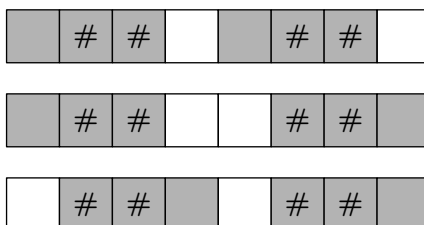


Рис. 1. Допустимые расстановки кораблей в начале игры.

Первый выстрел производится в клетку с номером 4. Это выстрел считается промахом, остаются допустимыми расстановки кораблей, приведенные на рис. 2. Теперь 5 клеток заведомо заняты.

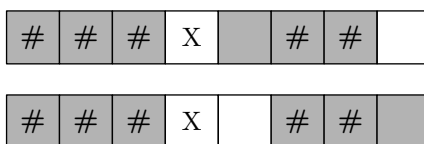


Рис. 2. Допустимые расстановки кораблей после первого выстрела.

Второй выстрел производится в клетку с номером 1, эта клетка не может быть заведомо занятой, поэтому игра завершается.

## Задача 7. Массовый прогноз

Имя входного файла: prediction.in  
Имя выходного файла: prediction.out  
Ограничение по времени: 1 секунда  
Ограничение по памяти: 256 МБ

В выборах председателя школьного клуба информатиков участвуют  $K$  кандидатов и  $N$  избирателей. Кандидаты пронумерованы от 1 до  $K$ , избиратели — от 1 до  $N$ .

По результатам голосования составляется список,  $i$ -й элемент этого списка равен номеру кандидата, за которого проголосовал  $i$ -й избиратель. Для каждого отрезка списка назначается наблюдатель, который подсчитывает голоса на этом отрезке. Таким образом, на выборах работают  $N(N+1)/2$  наблюдателей.

Если наблюдатель обнаружит кандидата, набравшего на его отрезке более половины голосов, он публикует в социальной сети прогноз о том, что этот кандидат победит в выборах.

Требуется написать программу, которая по списку голосов определяет количество опубликованных наблюдателями прогнозов.

### Формат входных данных

Первая строка входного файла содержит два числа  $N$  и  $K$  ( $1 \leq N \leq 500\,000$ ,  $1 \leq K \leq 500\,000$ ). Вторая строка содержит  $N$  чисел  $V_1, V_2, \dots, V_N$  — список голосов избирателей ( $1 \leq V_i \leq K$ ).

### Формат выходных данных

Выходной файл должен содержать единственное число — количество прогнозов.

### Примеры

prediction.in	prediction.out
5 2 1 2 1 2 1	9
3 7 5 2 6	3

### Система оценивания

Для окончательной проверки решений этой задачи используются 50 тестов. Тесты оцениваются независимо. Каждый тест оценивается в 2 балла. Значения  $N$  и  $K$  в тестах приведены в таблице.

В этой задаче результаты окончательной проверки во время тура недоступны.

Тест	N	K	Тест	N	K	Тест	N	K
1.	2	2	18.	2000	20	35.	90000	1000
2.	3	1	19.	3000	2000	36.	100000	5000
3.	5	5	20.	5000	2000	37.	125000	1
4.	10	10	21.	7500	200	38.	150000	12000
5.	20	2	22.	10000	10000	39.	150000	18
6.	30	3	23.	15000	1500	40.	200000	42000
7.	50	20	24.	20000	10	41.	250000	26000
8.	75	75	25.	25000	100	42.	300000	10000
9.	100	2000	26.	30000	15	43.	350000	102000
10.	150	30	27.	35000	35	44.	400000	12000
11.	200	50	28.	40000	10000	45.	450000	5000
12.	300	10	29.	45000	10000	46.	500000	2
13.	400	100	30.	50000	10000	47.	500000	102000
14.	500	2	31.	55000	13000	48.	500000	102000
15.	300	200	32.	60000	174	49.	500000	102000
16.	1000	2000	33.	70000	10000	50.	500000	501
17.	1500	100	34.	80000	1000			

## Задача 8. Флешмоб

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

Для участников олимпиады на главной площади города «У» планируется игра в форме флешмоба. Главная площадь замощена плитками, образующими клетчатое поле.

Сначала составляется план игры: каждый участник флешмоба получает номер в очереди выхода на площадь и координаты двух различных плиток, находящихся в одном ряду или столбце. После этого на площади раскладываются призы, затем участники выходят на площадь по очереди. Очередной участник забирает все призы, находящиеся в указанных ему клетках, и клетках, находящихся между ними.

Призы должны быть разложены так, чтобы каждому участнику достался по крайней мере один приз.

Требуется написать программу, которая по плану игры находит минимальное необходимое количество призов, и на какие именно плитки их следует разложить.

### Формат входных данных

В первой строке входного файла содержится число  $N$  — количество участников флешмоба ( $1 \leq N \leq 123\,456$ ). Каждая из последующих  $N$  строк содержит четыре целых числа  $x_{1i}, y_{1i}, x_{2i}, y_{2i}$  — координаты плиток для  $i$ -го участника ( $1 \leq x_{1i}, y_{1i}, x_{2i}, y_{2i} \leq 10^9$ ; либо  $x_{1i} = x_{2i}$ , либо  $y_{1i} = y_{2i}$ ). Участники перечислены в порядке выхода на площадь.

### Формат выходных данных

Первая строка выходного файла должна содержать число  $M$  — минимальное количество призов, которые должны быть разложены на площади. Каждая из последующих  $M$  строк должна содержать два числа  $px_i$  и  $py_i$  — координаты плитки, на которой должен лежать  $i$ -й приз.

Если вариантов размещения призов, удовлетворяющих условию задачи, несколько, то выведите любой из них. Если решения не существует, выведите единственное число 0.

### Система оценивания

Данная задача содержит три подзадачи. Для оценки каждой подзадачи используется своя группа тестов. Баллы за первые две подзадачи начисляются только в том случае, если все тесты из этой группы пройдены. Баллы за каждый тест третьей подзадачи выставляются независимо.

#### Подзадача 1

$N \leq 123$ . Все координаты не превосходят 234. Подзадача оценивается в 21 балл.

#### Подзадача 2

$N \leq 2543$ . Подзадача оценивается в 23 балла.

#### Подзадача 3

$N \leq 123\,456$ . Подзадача оценивается из 56 баллов.

### Примеры

<code>flashmob.in</code>	<code>flashmob.out</code>
5 2 1 2 4 2 4 4 4 5 1 1 1 4 4 4 2 4 2 1 2	5 1 2 4 3 1 1 3 4 2 3
3 1 1 1 3 2 1 2 3 1 2 2 2	0
4 1 1 1 3 2 1 2 3 3 3 3 1 1 3 4 3	4 4 3 3 1 2 1 1 1