## *Информация о результатах оценивания решений во время тура для всех задач*

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. Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте.

В каждой задаче можно задать, какое из прошедших предварительную проверку решений будет оцениваться. В этом случае баллы начисляются за лучшее решение из следующих:

- выбранного явно;
- последнего принятого на проверку решения.

Если выбор не сделан, то будет оцениваться лучшее решение из следующих:

- тех решений, по которым просмотрены баллы;
- последнего принятого на проверку решения.

# Задача 1. «Пароль»

| | |
|---|---|
| Имя входного файла: | `password.in` |
| Имя выходного файла: | `password.out` |
| Максимальное время работы на одном тесте: | 1 *секунда* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

Участник олимпиады разбирается с программой, которая шифрует пароль входа в систему. После работы эта программа выдает два натуральных числа, причем второе число получено из первого в результате замены некоторой непустой группы подряд идущих цифр первого числа на их сумму. Известно, что пароль – это группа цифр первого числа, замененная на их сумму во втором числе.

Требуется написать программу, которая по двум числам определяет номера позиций первой и последней цифры группы, являющейся искомым паролем.

## Формат входных данных

Входной файл содержит две строки. В первой строке записано первое число, состоящее не более чем из 100 000 цифр, во второй строке – второе число. Гарантируется, что числа не начинаются с нуля.

## Формат выходных данных

Выходной файл должен содержать два разделённых пробелом числа – номера позиций первой и последней цифры группы, которая была заменена в первом числе. Если решений несколько, можно вывести любое из них. Гарантируется, что решение существует.

## Примеры входных и выходных данных

| password.in | password.out |
|---|---|
| 2148<br>213 | 2 4 |
| 8<br>8 | 1 1 |
| 1223<br>1223 | 4 4 |
| 10002<br>1002 | 3 4 |

## Комментарий

В первом примере группа цифр 148 заменятся на число 13 = 1 + 4 + 8.

## Подзадачи и система оценки

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 пройдены.

### Подзадача 1 (оценивается в 30 баллов)

Первое число меньше $10^9$.

### Подзадача 2 (оценивается в 30 баллов)

Первое число меньше $10^{1000}$.

### Подзадача 3 (оценивается в 40 баллов)

Первое число меньше $10^{100\,000}$.

# Задача 2. «Вирусы и антивирусы»

| | |
|---|---|
| Имя входного файла: | `virus.in` |
| Имя выходного файла: | `virus.out` |
| Максимальное время работы на одном тесте: | 2 *секунды* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

Антивирусная IT-компания имеет официальную иерархическую структуру управления. В ней есть босс – единственный сотрудник, над которым нет начальника. Каждый из остальных сотрудников подчинён ровно одному сотруднику – своему начальнику. Начальник может иметь нескольких подчинённых и отдавать или передавать приказы любому из них. Приказы могут передаваться от одного сотрудника другому только по цепочке, каждый раз от начальника к его подчинённому.

Сотрудник *А главнее* сотрудника *Б* в этой иерархии, если *А* может отдать или передать приказ сотруднику *Б* непосредственно, или через цепочку подчинённых. Босс главнее любого сотрудника.

Оказалось, что все сотрудники объединены ещё в одну организованную подобным образом тайную иерархическую структуру, производящую компьютерные вирусы. В тайной структуре может быть другой босс, а у сотрудников – другие начальники.

Будем называть пару сотрудников *А* и *Б устойчивой*, если *А* главнее *Б* и в основной, и в тайной иерархических структурах.

Требуется написать программу, определяющую количество устойчивых пар в компании.

### Формат входных данных

В первой строке задано число $N$ – количество сотрудников компании ($1 \le N \le 100\,000$).

Во второй строке – $N$ целых чисел $a_i$, где $a_i = 0$, если в официальной иерархии сотрудник с номером $i$ является боссом, в противном случае $a_i$ равно номеру непосредственного начальника сотрудника номер $i$.

В третьей строке – $N$ целых чисел $b_i$, где $b_i = 0$, если в тайной иерархии сотрудник с номером $i$ является боссом, в противном случае $b_i$ равно номеру непосредственного начальника сотрудника номер $i$.

Нумерация сотрудников ведется с единицы в том порядке, в каком они упомянуты во входном файле.

### Формат выходных данных

Выходной файл должен содержать единственное число – количество устойчивых пар.

### Примеры входных и выходных данных

| virus.in | virus.out |
|---|---|
| 3<br>0 3 1<br>0 1 1 | 2 |
| 5<br>2 0 1 3 4<br>3 1 0 2 4 | 7 |

### Подзадачи и система оценки

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 пройдены.

#### Подзадача 1 (оценивается в 25 баллов)

Количество сотрудников $N$ не превосходит 100.

#### Подзадача 2 (оценивается в 25 баллов)

Количество сотрудников $N$ не превосходит 2000.

#### Подзадача 3 (оценивается в 50 баллов)

Количество сотрудников $N$ не превосходит 100 000.

# Задача 3. «Урюк»

| | |
|---|---|
| Имя входного файла: | `apricot.in` |
| Имя выходного файла: | `apricot.out` |
| Максимальное время работы на одном тесте: | 1 *секунда* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

В давние времена Золотая Орда ежегодно собирала дань золотыми монетами. Известный крымский хан Гирей решил схитрить: выплачивая дань из $N$ золотых монет, он подложил среди них одну фальшивую – более легкую монету. Об этом донесли казначею Золотой Орды. Для обнаружения подделки он решил использовать магические весы, работающие на урюке.

На чаши магических весов кладутся две кучи монет. Весы устанавливают, совпадает или различается вес этих куч. При этом, если кучи имеют разный вес, то весы указывают, какая из куч легче. При совпадении веса обеих куч весы требуют R плодов урюка, а при несовпадении – U плодов.

Казначей, сам любитель урюка, хочет и фальшивую монету обнаружить, и сэкономить на урюке.

Требуется написать программу, которая по заданному количеству монет $N$, при условии, что только одна из них легче других, укажет минимальное количество урюка, с помощью которого эта фальшивая монета гарантированно будет обнаружена.

### Формат входных данных

Во входном файле в единственной строке находятся три целых числа $N$, $R$ и $U$ ($2 \le N \le 1\,000\,000$, $1 \le R, U \le 1\,000\,000$), где $N$ – количество монет, $R$ – количество плодов урюка, затрачиваемых в случае совпадения веса куч монет, $U$ – количество плодов урюка, затрачиваемых в случае их различия. Все числа разделены пробелом.

### Формат выходных данных

Выходной файл должен содержать одно число – минимальное количество урюка, с помощью которого гарантированно будет обнаружена фальшивая монета.

### Пример входных и выходных данных

| apricot.in | apricot.out |
|---|---|
| 4 3 1 | 2 |
| 3 3 1 | 3 |
| 15 2 3 | 8 |
| 10 2 1 | 3 |

### Подзадачи и система оценки

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 пройдены.

#### Подзадача 1 (оценивается в 40 баллов)

$N$, $U$, $R$ не превосходят 200.

#### Подзадача 2 (оценивается в 30 баллов)

$N$, $U$, $R$ не превосходят 2000.

#### Подзадача 3 (оценивается в 30 баллов)

$N$, $U$, $R$ не превосходят 1 000 000.

# Задача 4. «Древний календарь»

| | |
|---|---|
| Имя входного файла: | `calendar.in` |
| Имя выходного файла: | `calendar.out` |
| Максимальное время работы на одном тесте: | 1 *секунда* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

Как известно, в 2012 году человечество с повышенным вниманием относится к древним календарям. Особый интерес представляют те из них, которые не заканчиваются 2012 годом. Потрясающее открытие в этом направлении сделано археологами Татарстана. В древних захоронениях они обнаружили прямоугольную табличку, которая после расшифровки сохранившихся знаков была записана в виде таблицы, состоящей из $N$ строк по $M$ десятичных цифр в каждой. Но полностью расшифровать табличку не удалось, так как некоторые цифры стерлись. Утраченные цифры в таблице были заменены символами «*».

По мнению археологов, найденная табличка представляет собой древний календарь, а записанные в ней $M$-значные числа являются номерами последовательных дней некоторого периода. Первое число является номером первого дня этого периода, а каждое следующее число на единицу больше предыдущего. По этому календарю конец света отсутствует, и после дня, обозначаемого с помощью $M$ девяток, следует номер дня из $M$ нулей.

Требуется написать программу, которая восстанавливает утраченные цифры так, чтобы число в каждой строке таблицы, начиная со второй, было на единицу больше предыдущего, и выводит номер первого дня в найденном календаре.

### *Формат входных данных*

В первой строке входного файла записаны натуральные числа $N$ и $M$ – количество строк в таблице и длина каждой строки соответственно ($1 \le N \le 100\,000$, $1 \le M \le 100\,000$, $M \times N \le 100\,000$). Далее следуют $N$ строк по $M$ символов в каждой, состоящих только из десятичных цифр от 0 до 9 и символов «*».

### *Формат выходных данных*

Выходной файл должен содержать одну строку, состоящую из $M$ цифр – номер первого дня календаря. Если вариантов восстановления несколько, можно вывести любой из них. Гарантируется, что хотя бы один способ восстановления существует.

### *Примеры входных и выходных данных*

| `calendar.in` | `calendar.out` |
|---|---|
| 1 2<br>23 | 23 |
| 3 3<br>1**<br>*1*<br>**1 | 109 |
| 2 3<br>9**<br>00* | 999 |
| 3 4<br>****<br>*0**<br>01** | 0098 |

## *Подзадачи и система оценки*

Данная задача содержит три подзадачи.

### *Подзадача 1 (оценивается в 40 баллов)*

$1 \le N \le 1000$, $1 \le M \le 100$. В этой подзадаче исходные данные таковы, что в каждом столбце таблицы есть, по крайней мере, одна сохранившаяся цифра. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы пройдены.

### *Подзадача 2 (оценивается из 30 баллов)*

$1 \le N \le 1000$, $1 \le M \le 100$. В этой подзадаче хотя бы один столбец содержит только символы «*». Каждый тест в этой подзадаче оценивается отдельно.

### *Подзадача 3 (оценивается в 30 баллов)*

$1 \le N \le 100\,000$, $1 \le M \le 100\,000$, $M \times N \le 100\,000$. Баллы за подзадачу начисляются только в том случае, если все тесты из этой группы пройдены.

## *Информация о результатах оценивания решений во время тура для всех задач*

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. Для каждой подзадачи сообщаются баллы за эту подзадачу и результат проверки программы на каждом тесте.

В каждой задаче можно задать, какое из прошедших предварительную проверку решений будет оцениваться. В этом случае баллы начисляются за лучшее решение из следующих:

- выбранного явно;
- последнего принятого на проверку решения.

Если выбор не сделан, то будет оцениваться лучшее решение из следующих:

- тех решений, по которым просмотрены баллы;
- последнего принятого на проверку решения.

# Задача 5. «Мозаика»

| | |
|---|---|
| Имя входного файла: | `mosaic.in` |
| Имя выходного файла: | `mosaic.out` |
| Максимальное время работы на одном тесте: | 1 *секунда* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

Все элементы магнитной мозаики фирмы «ABBYY» имеют прямоугольную форму. Два элемента можно соединить только в том случае, если у них совпадает хотя бы один из размеров: длина, ширина, или и то, и другое. Магнитные элементы поворачивать и переворачивать нельзя. Пару элементов мозаики, которые нельзя соединить, назовем *негармоничной*. Например, пара $1 \times 2$ и $2 \times 3$ является негармоничной, а пары $2 \times 3$ и $1 \times 3$ или $2 \times 3$ и $2 \times 3$ являются гармоничными.

Дизайнеры «ABBYY» выложили все элементы мозаики в ряд, не соединяя их между собой. Назовем *набором* несколько подряд лежащих элементов мозаики в этом ряду. Они выбрали несколько наборов элементов, которые хотят оставить для создания инсталляции. Для каждого такого набора им нужно выяснить, есть ли в нем негармоничная пара элементов.

Требуется написать программу, которая для различных наборов подряд лежащих элементов мозаики определит номера элементов, образующих негармоничную пару, или сообщит, что такой пары нет.

## Формат входных данных

В первой строке входного файла записано одно число $N$ – количество элементов, из которых состоит мозаика ($2 \le N \le 100\,000$). В следующих $N$ строках записаны по два целых числа $A_i$ и $B_i$, задающих длину и ширину $i$-го элемента мозаики соответственно ($1 \le A_i, B_i \le 10^9$, $1 \le i \le N$).

В $(N + 2)$-й строке записано одно целое число $K$ – количество наборов, в каждом из которых нужно определить номера двух негармоничных элементов ($1 \le K \le 100\,000$). В следующих $K$ строках записаны пары целых чисел $N_1$ и $N_2$ – номера первого и последнего элементов набора соответственно, в котором необходимо найти два негармоничных элемента мозаики ($1 \le N_1 < N_2 \le N$).

## Формат выходных данных

Выходной файл должен содержать $K$ строк, каждая из которых содержит два разделённых пробелом числа – номера элементов мозаики, образующих негармоничную пару в соответствующем наборе. Если решений несколько, можно вывести любое из них. Если в наборе негармоничная пара отсутствует, требуется вывести в соответствующей строке 0 0.

## Примеры входных и выходных данных

| mosaic.in | mosaic.out |
|---|---|
| 4<br>2 2<br>1 2<br>1 3<br>2 3<br>2<br>2 3<br>2 4 | 0 0<br>4 2 |

## Подзадачи и система оценки

Данная задача содержит четыре 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 (оценивается в 20 баллов)

Количество элементов мозаики $N \le 100$, число наборов $K \le 100$.

### Подзадача 2 (оценивается в 30 баллов)

Количество элементов мозаики $N \le 1\,000$, число наборов $K \le 1\,000$.

### Подзадача 3 (оценивается в 20 баллов)

Количество элементов мозаики $N \le 5\,000$, число наборов $K \le 5\,000$.

### Подзадача 4 (оценивается в 30 баллов)

Количество элементов мозаики $N \le 100\,000$, число наборов $K \le 100\,000$.

# Задача 6. «Театр начинается с актеров»

| | |
|---|---|
| Имя входного файла: | `theatre.in` |
| Имя выходного файла: | `theatre.out` |
| Максимальное время работы на одном тесте: | 1 *секунда* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

Участники олимпиады пришли в казанский театр на спектакль, где играют $N$ неизвестных для них актеров. В фойе театра висят портреты всех актеров труппы, которая в полном составе задействована в спектакле. Портреты не подписаны. Зрителям раздали программки, в которых для каждого действия спектакля приводится список фамилий участвующих в нем актеров, но не указаны их роли.

Театрал Виталий решил узнать, как выглядит каждый из актеров, упомянутых в программке. Для этого в антракте после каждого действия он выходил в фойе и сопоставлял портреты с увиденными актерами.

Требуется написать программу, которая по заданному числу актеров $N$ и списку фамилий актеров, участвующих в каждом из $M$ действий, определяет номер действия, после которого впервые становится возможным установить соответствие между фамилией актера из программки и его портретом.

## Формат входных данных

Первая строка входного файла содержит два натуральных числа $N$ – число актеров и $M$ – количество действий в спектакле ($1 < N \le 100\,000$, $1 \le M \le 100\,000$). В каждой из следующих $M$ строк сначала записано количество актеров $K_i$, участвующих в $i$–ом действии ($1 \le K_i \le N$, $K_1 + K_2 + \ldots + K_M \le 100\,000$), а затем $K_i$ различных натуральных чисел, не превосходящих $N$, обозначающих фамилии этих актеров. Соседние числа в каждой строке разделены пробелом.

## Формат выходных данных

Выходной файл должен содержать одну строку, состоящую из $N$ записанных через пробел чисел. $i$-е число этой строки – это номер действия, после которого впервые становится возможным установить соответствие между $i$–м актером и его портретом. Если к концу спектакля установить соответствие между каким-либо актером и его портретом так и не удалось, то соответствующее число в строке должно быть равно нулю.

## Примеры входных и выходных данных

| theatre.in | theatre.out |
|---|---|
| 3 3<br>2 1 2<br>2 3 2<br>2 1 2 | 2 2 1 |
| 5 3<br>3 1 2 3<br>3 2 3 1<br>2 1 3 | 0 3 0 0 0 |
| 4 3<br>1 1<br>1 3<br>1 2 | 1 3 2 3 |

## Комментарий

В первом примере три актера участвуют в спектакле с тремя действиями. В первом действии участвуют два актера с номерами 1 и 2. Так как актеров всего трое, то после первого акта становится понятно, какой портрет соответствует актеру с номером 3, поэтому третье число строки выходного файла равно 1.

Во втором действии участвуют два актера с номерами 3 и 2. Поскольку только второй актер участвовал и в первом, и во втором действиях, то его портрет можно определить после второго действия. А так как портретов всего три, то после второго действия можно установить, что последний портрет соответствует актеру номер 1. Третье действие на ответ не влияет.

## Подзадачи и система оценки

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 пройдены.

### Подзадача 1 (оценивается в 30 баллов)

Количество актеров $N$ не превосходит 100, количество действий $M$ не превосходит 100, $K_1 + K_2 + \ldots + K_M \leq 100$.

### Подзадача 2 (оценивается в 30 баллов)

Количество актеров $N$ не превосходит 10000, количество действий $M$ не превосходит 10000, $K_1 + K_2 + \ldots + K_M \leq 10000$.

### Подзадача 3 (оценивается в 40 баллов)

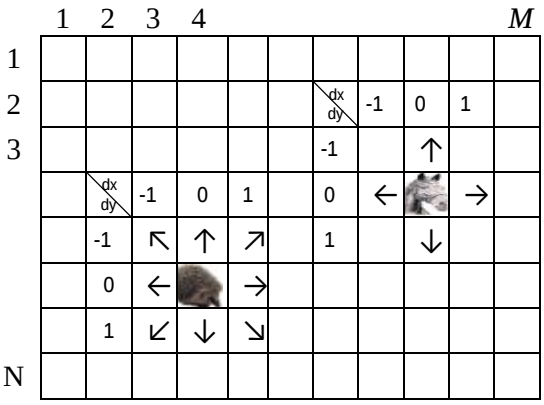
Количество актеров $N$ не превосходит 100000, количество действий $M$ не превосходит 100000, $K_1 + K_2 + \ldots + K_M \leq 100000$.

# Задача 7. «Ёжик в тумане»

| | |
|---|---|
| Имя входного файла: | *стандартный ввод* |
| Имя выходного файла: | *стандартный вывод* |
| Максимальное время работы на одном тесте: | 1 *секунда* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

*Над рекой поднялся туман, и грустная белая Лошадь утонула в нем по грудь.*
*– Вот, – сказал Ёжик. – Ничего не видно. И даже лапы не видно.*
*– Лошадь! – позвал он. Но лошадь ничего не сказала. "Где же Лошадь?" - подумал Ёжик.*
*(С.Г. Козлов)*

Ёжик спустился в туман и оказался в прямоугольной долине размером $N$ на $M$ метров, по которой бродит Лошадь. Ёжик хочет ее найти. Будем считать, что в каждый момент времени и Ёжик, и Лошадь находятся в одной из $N \times M$ клеток. Туман настолько густой, что Лошадь не видно, даже если она находится в той же самой клетке , что и Ёжик. К счастью Ёжик обладает очень острым слухом и может понять, в каком направлении сместилась Лошадь. Он также может позвать Лошадь, и, если она находится в одной клетке с Ёжиком, то Лошадь его услышит и обязательно отзовется.




В каждый момент времени Ёжик может сместиться в соседнюю клетку по горизонтали, вертикали или диагонали. Потом он отчетливо слышит, куда сместилась Лошадь относительно своего старого местоположения. Лошадь за единицу времени смещается на одну клетку только по горизонтали или вертикали (влево, вверх, вправо или вниз). При этом Лошадь не выходит за границы долины, поэтому и Ёжик не должен этого делать.

Требуется написать программу, которая поможет Ёжику, знающему свое начальное положение и следящему за передвижениями Лошади, как можно быстрее ее найти.

Это интерактивная задача. В процессе тестирования программа-решение будет взаимодействовать с использованием стандартных потоков ввода/вывода с программой, моделирующей поведение лошади.

### *Протокол взаимодействия программ*

Сначала программа-решение должна прочитать из стандартного потока ввода натуральные числа $N$ и $M$, записанные в первой строке, а из второй строки координаты начального местоположения Ёжика – два натуральных числа: $x_0$ – номер столбца, $y_0$ – номер строки ($1 \le x_0 \le M$, $1 \le y_0 \le N$). Числа в каждой строке разделены пробелом.

Затем программа-решение начинает взаимодействие с программой, моделирующей поведение лошади, в соответствии со следующим протоколом:

1. Программа выводит в стандартный поток вывода одну строку, описывающую ход Ёжика, которая содержит три числа: его перемещение в виде указания смещения по горизонтали $dx$ ($dx = -1$, 0 или 1) и по вертикали $dy$ ($dy = -1$, 0 или 1), а также число 1, если Ёжик зовет Лошадь в клетке, в которую он при этом попадет, или 0 – если не зовет. Вывод должен завершаться переводом строки и сбросом буфера потока вывода. Для этого используйте
   - `flush(output)` в паскале или Delphi;
   - `fflush(stdout)` или `cout.flush()` в C/C++;
   - `Console.out.flush()` в Visual Basic.
2. После этого программа должна считать из стандартного потока ввода ответ программы, сообщающей о действии Лошади. Ответ состоит из трех чисел, расположенных в одной строке через пробел. Первое число ответа может быть равно 0 или 1, где
   - 0 означает, что Ёжик не пытался позвать Лошадь либо позвал, но Лошади в его клетке нет. В этом случае следующие два числа обозначают очередное смещение Лошади по горизонтали $dx$ ($dx = -1$, 0 или 1) и по вертикали $dy$ ($dy = -1$, 0 или 1), при этом хотя бы одно из значений $dx$ или $dy$ равно нулю;

- 1 означает, что Ёжик позвал Лошадь, и она действительно оказалась в той же клетке, что и он. В этом случае другие два числа равны 0, и программа-решение должна закончить свою работу.

Программа-решение не должна делать более 10 000 ходов.

### Пример взаимодействия

| стандартный ввод | стандартный вывод |
|---|---|
| 2 3<br>1 2<br>0 1 0<br>0 0 0<br>1 0 0 | <br>0 0 1<br>1 -1 0<br>1 0 1 |

### Комментарий

Ёжик находился в клетке (1, 2). Сначала он попробовал позвать Лошадь в той же клетке (вывод: 0 0 1), но Лошади там не оказалось, и она сместилась вправо (ввод: 0 1 0). Ёжик сместился по диагонали, но Лошадь звать не стал (вывод: 1 –1 0), а Лошадь осталась на месте (ввод: 0 0 0). Ежик сместился вправо и позвал Лошадь (вывод: 1 0 1). Лошадь оказалась в той же клетке и отозвалась (ввод: 1 0 0). Значит, изначально Лошадь находилась в клетке (2, 1), а встретились они в клетке (3, 1). Ёжик при этом сделал три хода и дважды запросил местоположение Лошади.

### Подзадачи и система оценки

В данной задаче две подзадачи. Каждый тест в обеих подзадачах оценивается отдельно. Оценка за тест вычисляется по формуле $\min\{10, \text{round}(10\times(J/S)^2)\}$, где 10 – оценка в баллах за тест, $S$ – количество ходов, которое потребовалось программе-решению, чтобы обнаружить Лошадь, $J$ – количество ходов, которое требуется заданному эталонному решению при том же начальном положении Ёжика. Округление ведется по правилам математики.

#### Подзадача 1 (оценивается из 40 баллов)

$2 \le N, M \le 10$.

#### Подзадача 2 (оценивается из 60 баллов)

$2 \le N, M \le 30$. В этой подзадаче количество запросов о том, есть ли Лошадь в текущей клетке, не должно превышать $N\times M$.

### Вспомогательная программа

Для тестирования своего решения вы можете использовать вспомогательную программу «`runpair`», которая находится у вас в каталоге «`c:\work\runpair`».

Она позволяет запустить две программы и перенаправить стандартный поток вывода первой программы на стандартный поток ввода второй программы и наоборот.

Для тестирования с ее помощью программы-решения вам придется помимо программы-решения написать программу, моделирующую поведение лошади. Тогда можно запустить одновременно программу, моделирующую поведение лошади, и программу-решение, с помощью команды

`runpair` «`исполнимый файл лошади`» «`исполнимый файл ёжика`»

и на экране будет отображен их диалог.

# Задача 8. «Ордынское войско»

| | |
|---|---|
| Имя входного файла: | `army.in` |
| Имя выходного файла: | `army.out` |
| Максимальное время работы на одном тесте: | 1 *секунда* |
| Максимальный объем используемой памяти: | 256 *мегабайт* |
| Максимальная оценка | 100 *баллов* |

Готовясь к бою, хан Гирей пронумеровал всех воинов своего войска натуральными числами от 1 до $N$. Поскольку воины умеют сражаться, но не умеют считать, при любом построении в шеренгу они выстраиваются в произвольном порядке.

Одного или несколько воинов, стоящих в шеренге, будем называть *отрядом*. Отряд назовем *правильным*, если номера этих воинов в том порядке, в котором они стоят в шеренге, образуют упорядоченную по возрастанию последовательность чисел. Среди всех правильных отрядов хан Гирей выбирает *ударный отряд* – самый большой по количеству воинов. Так, в шеренге 1 3 2 4 из четырех воинов ударными являются отряды 1 3 4 и 1 2 4, а отряд 1 4 – один из правильных, но не ударный.

Некоторые воины являются личными телохранителями хана Гирея.

Требуется составить программу, определяющую количество таких шеренг, в которых телохранители хана образуют ударный отряд.

## Формат входных данных

В первой строке входного файла задано натуральное число $N$ – общее количество воинов ($1 \le N \le 15$). Во второй строке задано натуральное число $K$ – количество телохранителей хана ($1 \le K \le N$). В третьей строке через пробел указаны $K$ различных натуральных чисел, не превосходящих $N$, – номера телохранителей хана в порядке возрастания.

## Формат выходных данных

Выходной файл должен содержать единственное число – количество различных расстановок всех воинов в шеренгу так, чтобы все телохранители хана были ударным отрядом в каждой из таких расстановок.

## Примеры входных и выходных данных

| army.in | army.out |
|---|---|
| 5<br>3<br>1 3 4 | 11 |
| 3<br>3<br>1 2 3 | 1 |
| 1<br>1<br>1 | 1 |

## Комментарий

В первом примере войско состоит из пяти воинов. Ударный отряд должен состоять из трех воинов с номерами 1, 3 и 4. Этому условию удовлетворяют следующие 11 шеренг: (1, 3, 2, 5, 4), (1, 3, 5, 2, 4), (1, 3, 5, 4, 2), (1, 5, 3, 2, 4), (1, 5, 3, 4, 2), (2, 1, 3, 5, 4), (2, 1, 5, 3, 4), (2, 5, 1, 3, 4), (5, 1, 3, 2, 4), (5, 1, 3, 4, 2), (5, 2, 1, 3, 4).

## *Подзадачи и система оценки*

Данная задача содержит семь подзадач. 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 (оценивается в 40 баллов)*

$1 \le N \le 8$.

### *Подзадача 2 (оценивается в 10 баллов)*

$9 \le N \le 10$.

### *Подзадача 3 (оценивается в 10 баллов)*

$N = 11$.

### *Подзадача 4 (оценивается в 10 баллов)*

$N = 12$.

### *Подзадача 5 (оценивается в 10 баллов)*

$N = 13$.

### *Подзадача 6 (оценивается в 10 баллов)*

$N = 14$.

### *Подзадача 7 (оценивается в 10 баллов)*

$N = 15$.