# Задача A. Перевёрнутый порядок

| | |
|---|---|
| Имя входного файла: | another.in |
| Имя выходного файла: | another.out |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мебибайт |

На вокзал Города Роботов прибыл товарный поезд. Поезд состоит из паровоза и $n$ вагонов-платформ. На каждой платформе закреплён один контейнер. Каждый контейнер обозначен заглавной буквой английского алфавита.

Робот-грузчик хочет разгрузить этот поезд. Ожидалось, что контейнеры будут уложены на платформах последовательно: контейнер на первой платформе (той, которая ближе всего к паровозу) будет обозначен буквой «A», контейнер на второй — буквой «B», и так далее. Робот-грузчик получил программу, которую нужно выполнить при такой расстановке, чтобы разгрузить поезд и отправить каждый контейнер по назначению.

По прибытии поезда неожиданно выяснилось, что порядок контейнеров — перевёрнутый: на последней (самой далёкой от паровоза) платформе стоит контейнер, обозначенный буквой «A», на предпоследней — обозначенный буквой «B», и так далее. Переделывать на ходу всю программу разгрузки — опасно: ведь в неё могут закрасться ошибки. Поэтому робот-грузчик решил сначала расставить контейнеры на платформах в порядке, который ожидался при составлении полученной им программы («A» на первой платформе, «B» на второй, и так далее), а затем выполнить её без изменений.

Робот-грузчик собирается переупорядочивать контейнеры, делая *простые перестановки*. Одна простая перестановка делается так: выбираются два различных контейнера и меняются местами. Конечно, робот-грузчик хочет сделать минимально возможное количество простых перестановок, которое позволит расставить контейнеры в нужном порядке.

Ваша задача — выяснить, какое минимальное количество простых перестановок придётся сделать, а также составить план: какие именно простые перестановки и в какой очерёдности делать, чтобы из перевёрнутого порядка контейнеров получить ожидаемый порядок.

## Формат входных данных

В первой строке ввода задано целое число $n$ — количество платформ ($0 \leqslant n \leqslant 26$).

## Формат выходных данных

В первой строке выведите $k$ — минимальное количество простых перестановок двух контейнеров, которое нужно сделать, чтобы привести контейнеры в ожидаемый порядок. В каждой из следующих $k$ строк выведите сами простые перестановки, по одной на строке, в той очерёдности, в котором их следует выполнять. Каждая простая перестановка должна быть задана двумя различными заглавными буквами английского алфавита через пробел — обозначениями на тех контейнерах, которые нужно поменять местами. Напомним порядок букв в английском алфавите: «A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z».

Если правильных ответов несколько, можно вывести любой из них.

## Примеры

| another.in | another.out |
|---|---|
| 3 | 1<br>A C |
| 4 | 2<br>B C<br>D A |

## Пояснение к примерам

В первом примере из порядка контейнеров «C, B, A» нужно сделать порядок «A, B, C». Для этого хватит одной простой перестановки: поменять местами контейнеры «A» и «C».

Во втором примере из порядка контейнеров «D, C, B, A» нужно сделать порядок «A, B, C, D». Тут уже придётся сделать как минимум две простых перестановки: например, сначала поменять местами «B» и «C», а затем — «D» и «A».

# Задача B. Мыльные пузыри

| | |
|---|---|
| Имя входного файла: | `bubbles.in` |
| Имя выходного файла: | `bubbles.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мебибайт |

В парках человеческих городов нередко устанавливаются фонтаны. В парке Города Роботов роль фонтана играет Генератор Мыльных Пузырей. Это устройство, будучи включённым, в начале каждой секунды выпускает в воздух над собой $a$ мыльных пузырей.

Воздушная среда вокруг Генератора устроена так, что в конце каждой секунды половина пузырей, находящихся в воздухе, лопается; если пузырей было нечётное количество, половина округляется вверх. Например, если в какую-то секунду в воздухе находилось $100$ пузырей, в конце этой секунды $50$ из них лопаются; если пузырей было $75$, лопается $38$ из них.

В этом году роботы решили не выключать Генератор на зиму, как они обычно делали, а поставить вокруг него шатёр на время холодов. Чтобы узнать оптимальные размеры шатра, роботы хотят выяснить, каково максимальное количество пузырей, которые могут одновременно оказаться в воздухе после того, как Генератор будет включён. Также они хотят узнать, на какой по счёту секунде после включения Генератора в воздухе впервые окажется такое количество пузырей.

Ваша задача — помочь роботам определить эти числа. Считайте, что до включения Генератора в воздухе над ним нет ни одного пузыря. Роботы гарантируют, что количество пузырей в воздухе не будет расти до бесконечности.

## Формат входных данных

В первой строке ввода задано целое число $a$ — количество пузырей, которые выпускает Генератор Мыльных Пузырей в начале каждой секунды ($1 \leqslant a \leqslant 10\,000$).

## Формат выходных данных

В первой строке выведите два числа, разделив их пробелом. Первым выведите максимальное количество пузырей, которые могут одновременно оказаться в воздухе после того, как Генератор Мыльных Пузырей будет включён. Вторым — порядковый номер секунды после включения Генератора, на которой в воздухе впервые окажется такое количество пузырей. Секунды нумеруются с единицы.

## Примеры

| bubbles.in | bubbles.out |
|---|---|
| 2 | 3 2 |
| 3 | 5 3 |

## Пояснение к примерам

Вычислим, сколько пузырей находится в воздухе в начале и в конце каждой секунды.

В первом примере:
- в начале первой секунды 2 пузыря, в конце 1;
- в начале второй секунды 3 пузыря, в конце 1;
- в начале третьей секунды 3 пузыря, в конце 1;

и так далее.
Максимальное количество пузырей: 3. Впервые оно достигается на второй секунде.

Во втором примере:
- в начале первой секунды 3 пузыря, в конце 1;
- в начале второй секунды 4 пузыря, в конце 2;
- в начале третьей секунды 5 пузырей, в конце 2;
- в начале четвёртой секунды 5 пузырей, в конце 2;

и так далее.
Максимальное количество пузырей: 5. Впервые оно достигается на третьей секунде.

# Задача C. Подъёмник

| | |
|---|---|
| Имя входного файла: | `crane.in` |
| Имя выходного файла: | `crane.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мебибайт |

Если посмотреть сбоку на склад Города Роботов, его пол, стены и крыша выглядят как отрезки прямых, а стены образуют с полом прямой угол. Длина склада равна $a$ метрам. Левая стена имеет высоту $p$ метров, а правая — высоту $q$ метров. (Если $p = q$, склад сбоку выглядит как прямоугольник, а в общем случае — как прямоугольная трапеция.)

На этом складе есть передвижной подъёмник. Сбоку он выглядит как вертикальный отрезок прямой. Нижний конец подъёмника находится на полу, а на верхнем конце установлен счётчик — прибор для измерения пройденного расстояния.

Счётчик был недавно доставлен на склад, и роботы хотят проверить его при помощи подъёмника. Для этого проводится следующий эксперимент. Подъёмник движется равномерно и прямолинейно слева направо по складу с постоянной скоростью $u$ метров в секунду. Одновременно с этим подъёмник перемещает счётчик либо вверх со скоростью $v$ метров в секунду, либо вниз с той же по величине скоростью $v$ метров в секунду. Изначально подъёмник установлен вплотную к левой стене, а счётчик располагается на уровне пола и движется вверх. Как только счётчик касается потолка или пола, направление движения по вертикали меняется на противоположное; при этом скорость движения слева направо по складу не изменяется. Движение заканчивается, когда подъёмник касается правой стены склада.

Ваша задача — найти фактическое расстояние, которое прошёл счётчик, чтобы проверить его показания. Толщина стен, пола и потолка, а также размеры подъёмника и счётчика настолько малы, что ими в этой задаче следует пренебречь. Напомним, что расстояние между двумя точками $(x_1, y_1)$ и $(x_2, y_2)$ по прямой равно $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

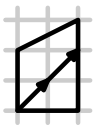
## Формат входных данных

В первой строке ввода заданы через пробел целые числа $a$, $p$ и $q$ — длина склада, высота левой стены и высота правой стены в метрах ($1 \leqslant a, p, q \leqslant 100$). Во второй строке заданы через пробел целые числа $u$ и $v$ — скорость движения подъёмника слева направо и скорость движения счётчика вверх или вниз в метрах в секунду ($1 \leqslant u, v \leqslant 100$).

## Формат выходных данных

В первой строке выведите одно число — фактическое расстояние, пройденное счётчиком. Выводите вещественные числа как можно точнее! Ваш ответ должен отличаться от точного не больше, чем на $0.001$.

## Примеры

| crane.in | crane.out | Пояснение |
|---|---|---|
| 2 2 3<br>1 1 | 2.828427125 |  |
| 7 2 2<br>3 2 | 8.412952976 |  |

## Пояснение к примерам

В первом примере счётчик движется равномерно и прямолинейно. Пройденное расстояние между точками $(0, 0)$ и $(2, 2)$ равно $\sqrt{2^2 + 2^2} = \sqrt{8} = 2.828427125...$.

Во втором примере счётчик сначала движется из начала координат в точку $(3, 2)$, затем меняет вертикальную скорость на противоположную и перемещается в $(6, 0)$, после чего опять меняет вертикальную скорость на противоположную и движется до точки $(7, \frac{2}{3})$. Длины отрезков пути складываются в суммарную длину так: $3.605551275... + 3.605551275... + 1.201850425... = 8.412952976...$.

# Задача D. Делитель

| | |
|---|---|
| Имя входного файла: | `divider.in` |
| Имя выходного файла: | `divider.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мебибайт |

В музее Города Роботов есть странное устройство: Делитель. Этому устройству можно задать набор различных целых положительных чисел. После этого Делитель находит и предъявляет все вещественные числа, которые имеют вид $x/y$, где $x$ и $y$ — числа из заданного набора (возможно, $x = y$).

Использование устройства осложняется тем, что оно работает не для любого набора чисел, а только для *базисов*. Набор называется базисом, если никакое из чисел $z$ этого набора нельзя представить в виде $x/y$, где $x$ и $y$ — также числа из этого набора (это должно быть верно, даже если какие-то числа из этой тройки совпадают). У каждого базиса есть *размер* — количество элементов в нём.

Желая произвести впечатление на посетителей музея, робот-экскурсовод хочет ввести базис как можно большего размера. Однако с устройством ввода чисел в Делитель тоже не так-то просто работать. Например, сегодня оно принимает только целые числа от 1 до $n$.

Ваша задача — найти базис как можно большего размера изо всех базисов, которые содержат только целые числа от 1 до $n$.

## Формат входных данных

В первой строке ввода задано целое число $n$ — максимальное целое число, которое сегодня можно ввести в Делитель ($1 \leqslant n \leqslant 100$).

## Формат выходных данных

В первой строке выведите $k$ — максимальный размер базиса, в который могут входить только целые числа от 1 до $n$. Во второй строке выведите сам базис такого размера: $k$ различных целых чисел от 1 до $n$, разделённых пробелами.

Если правильных ответов несколько, можно вывести любой из них.

## Примеры

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

## Пояснение к примерам

В первом примере базис может содержать только числа 1 и 2. Взять число 1 нельзя, поскольку $1/1 = 1$. Остаётся взять число 2.

Во втором примере из чисел 3 и 2 можно составить базис: ведь числа $2/2 = 1$, $3/3 = 1$, $3/2 = 1.5$ и $2/3 = 0.666666...$ в нём не содержатся. Существует и другой правильный ответ: числа 3 и 4 также образуют базис. Базис из трёх и более элементов построить не удастся, поскольку из трёх чисел 1, 2 и 4 в базис нельзя взять никакие два одновременно.

# Задача E. Исправление адресов

| | |
|---|---|
| Имя входного файла: | `error.in` |
| Имя выходного файла: | `error.out` |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мебибайт |

В почтовой системе Города Роботов адреса записываются как последовательности из девяти десятичных цифр (000 000 000 – 999 999 999). Всего в городе $n$ различных существующих адресов, по которым могут быть доставлены письма.

В главное почтовое отделение Города Роботов прибыли $m$ конвертов. На каждом из конвертов написан адрес, также состоящий из девяти десятичных цифр. Однако некоторые адреса могут быть написаны с ошибками, а некоторые конверты даже могли попасть в почтовое отделение города по ошибке.

Ваша задача — восстановить правильное написание адресов. Считается, что адрес $s$ на конверте *можно исправить* на адрес $t$ в Городе Роботов, если $s$ и $t$ либо совпадают, либо различаются ровно в одной цифре из девяти. Для каждого конверта, адрес на котором исправляется однозначно, выведите исправленный адрес, по которому нужно доставить этот конверт. Если адрес на конверте нельзя исправить ни на какой адрес в городе, поменяв не более чем одну цифру — или, наоборот, существует больше одного адреса в городе, на который его можно исправить — следует также это выяснить.

## Формат входных данных

В первой строке ввода задано целое число $n$ — количество различных адресов в Городе Роботов ($1 \leqslant n \leqslant 100$). В следующих $n$ строках записаны сами адреса, по одному на строке. Каждый адрес состоит ровно из девяти десятичных цифр без пробелов.

Следующая строка ввода содержит целое число $m$ — количество конвертов ($1 \leqslant m \leqslant 100$). В следующих $m$ строках записаны адреса, указанные на конвертах, по одному на строке. Каждый адрес также состоит ровно из девяти десятичных цифр без пробелов.

Имейте в виду, что, хотя все $n$ адресов, существующих в Городе Роботов, различны, на нескольких или даже всех $m$ конвертах может быть написан один и тот же адрес.

## Формат выходных данных

Выведите $m$ строк. В $i$-й из этих строк выведите адрес в Городе Роботов, по которому необходимо доставить $i$-й конверт. Помните, что адрес должен состоять ровно из девяти десятичных цифр без пробелов. Если подходящих адресов нет, выведите вместо адреса число `-1`. Если же подходящих адресов больше одного, выведите вместо адреса число `-2`.

## Примеры

| error.in | error.out |
|---|---|
| 2<br>000000000<br>999999999<br>3<br>012345678<br>999999989<br>000000000 | -1<br>999999999<br>000000000 |

| error.in | error.out |
|---|---|
| 3<br>123123123<br>124123123<br>134123123<br>2<br>124123123<br>135123123 | -2<br>134123123 |

## Пояснение к примерам

В первом примере (слева):

- в первом адресе на конверте пришлось бы поменять почти все цифры, чтобы получить какой-то из существующих в городе адресов;
- во втором адресе нужно поменять одну цифру;
- в третьем адресе вообще ничего менять не нужно.

Во втором примере (справа):

- первый адрес на конверте может быть исправлен на все три существующих адреса;
- второй адрес восстанавливается однозначно.