Задача A. Растояние Хэмминга

Имя входного файла:	`hamming.in`
Имя выходного файла:	`hamming.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

В связи с особенностями линии связи, используемой для передачи сообщений из пункта A в пункт B, каждый бит принятого сообщения с вероятностью 0.001 содержит ошибку.

Из пункта A в пункт B было послано одно из n сообщений m_1, m_2, ..., m_n. В пункте B было принято сообщение s.

Ваша задача заключается в определении наиболее вероятного исходного сообщения. Очевидно, что оно будет одним из тех сообщений, *расстояние Хэмминга* между которым и строкой s минимально.

Расстоянием Хэмминга двух строк a и b одинаковой длины называется количество позиций, в которых эти строки различаются (количество элементов в множестве $\{i | 1 \le i \le |a|, a_i \ne b_i\}$).

Формат входного файла

Первая строка входного файла содержит s — принятое сообщение. Вторая строка содержит целое число n — количество сообщений, которые могли быть отправлены. Следующие n строк содержат m_i — эти сообщения.

Длины всех сообщений равны ($|s| = |m_1| = |m_2| = ... = |m_n|$). Сообщения состоят только из символов *0* и *1*.

Размер входного файла не превосходит 256Кб.

Формат выходного файла

В первую строку выходного файла выведите k — количество сообщений, на которых достигается минимум расстояния Хэмминга. Во вторую строку выведите в порядке возрастания k чисел — номера этих сообщений.

Примеры

hamming.in	**hamming.out**
010101 3 110011 011001 000111	2 2 3

Задача B. Клавиатура

Имя входного файла:	`keyboard.in`
Имя выходного файла:	`keyboard.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Для данной буквы латинского алфавита нужно вывести справа стоящую букву на стандартной клавиатуре. При этом клавиатура замкнута, т.е. справа от буквы «p» стоит буква «a», от буквы «l» стоит буква «z», а от буквы «m» — буква «q».

Формат входного файла

Файл содержит один символ — маленькая букву латинского алфавита.

Формат выходного файла

Вывести букву стоящую справа от заданной буквы, с учетом замкнутости клавиатуры.

Примеры

keyboard.in	keyboard.out
q	w
w	e
e	r
y	u
v	b
u	i
r	t
b	n
i	o
o	p
p	a
a	s
d	f
j	k
f	g
g	h
h	j
k	l
l	z
s	d
t	y
z	x
x	c
c	v
n	m
m	q

Задача C. Маски подсетей

Имя входного файла:	`mask.in`
Имя выходного файла:	`mask.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Рассмотрим компьютерную сеть с настроенной TCP/IP маршрутизацией. Будем рассматривать некоторую ее модификацию. А именно в этой сети находить N подсетей. Каждая подсеть характеризуется своей маской. Маска подсети представляет собой 4 одно байтных числа, разделенных точкой. Причем для масок выполнено следующее свойство: если представить маску в двоичном виде, то сначала она будет содержать k единиц, а потом q нулей, причем $k + q = 32$. Например 255.255.255.0 — маска подсети, а 192.168.0.1 — нет.

Поясним, как получается двоичное представление IP-адреса. Для этого числа, составляющие IP-адрес, представляются в двоичной системе счисления (при этом каждое из них дополняется ведущими нулями до длины в 8 цифр), после чего удаляются точки. Получившееся 32-битное число и есть двоичное представление IP-адреса. Например, для адреса 192.168.0.1 этот процесс выглядит так: 192.168.0.1 $\rightarrow$ 11000000.10101000.00000000.00000001 $\rightarrow$ 11000000101010000000000000000001. Таким образом, двоичным представлением IP-адреса 192.168.0.1 является 11000000101010000000000000000001.

Будем говорить, что два компьютера с IP_1 и IP_2 лежат в подсети, если $IP_1 \wedge Mask = IP_2 \wedge Mask$, где $Mask$ — маска этой подсети, а $\wedge$ — операция побитового логического «и». IP компьютера представляет собой так же 4 одно байтных числа, разделенных точкой.

Вам даны M пар IP адресов компьютеров. Для каждой из них Вам надо определить, в скольких подсетях из заданных они лежат.

Формат входного файла

В первой строке входного файла записано число N — количество подсетей. В следующих N строках перечислены маски этих подсетей. В $N + 2$ строке находится число M ($0 \le M \le 100000$). В следующих M строках записаны пары IP адресов, разделенных пробелом.

Все заданные маски подсетей различны.

Формат выходного файла

Для каждой пары IP адресов в отдельной строке выходного файла выведите количество подсетей в которых лежат оба компьютера.

Примеры

mask.in	mask.out
2 255.255.255.255 255.255.255.0 3 192.168.31.1 192.168.31.2 192.168.31.3 192.168.31.4 192.168.31.1 192.167.31.2	1 1 0

Первая и вторая пара IP адресов лежит только в первой подсети. Третья пара не лежит ни в первой, ни во второй подсети.

Задача D. Дом — Школа — Дом

Имя входного файла:	`metro.in`
Имя выходного файла:	`metro.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Мальчик Вася каждый день ездит на метро. Утром он едет в школу, а вечером того же дня, обратно из школы, домой. Для того, чтобы немного сэкономить, он покупает электронную смарт-карту на X поездок. Когда он хочет зайти в метро, он прикладывает карту к турникету. Если на карте осталось ненулевое количество поездок, то турникет пропускает Васю и списывает с карты одну поездку. Если же на карте не осталось поездок, то турникет не пропускает Васю, и он (Вася) вынужден купить на этой же станции новую карту на X поездок и вновь пройти через турникет.

Вася заметил, что в связи с тем, что утром метро переполнено, покупать новую карту утром накладно по времени и он может опоздать в школу. В связи с этим он хочет понять: будет ли такой день, что с утра, поехав в школу, окажется, что у него на карточке ноль поездок.

Вася больше никуда на метро не ездит и поэтому заходит в метро только на станции около дома и на станции около школы.

Формат входного файла

Во входном файле содержиться ровно 2 строки. В первой содержится слово «`School`» или «`Home`» в зависимости от того, где первый раз Вася купил карточку на X поездок. Во второй строке сожержиться натуральное число $X, 1 \leq X \leq 1000$.

Формат выходного файла

Выведете слово «`Yes`», если будет такой день, что утром у Васи на карточке окажется ноль поездок.

Примеры

metro.in	metro.out
Home 1	Yes
School 2	No

Задача E. N-угольное колесо

Имя входного файла:	`nwheel.in`
Имя выходного файла:	`nwheel.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

На одном известном автозаводе страны N-мерики главный инженер-рационализатор внес предложение вместо круглых колес использовать колеса в форме правильных N-угольников. "При этом", — сказал он, "важным показателем качества такого колеса будет разность между радиусом описанной окружности и радиусом вписанной окружности."

Задано число N и длина A стороны N-угольного колеса. Найдите описанный главным инженером-рационализатором показатель качества колеса.

Формат входного файла

Входной файл содержит два натуральных числа: N и A $(3 \le N \le 1000, 1 \le A \le 1000)$.

Формат выходного файла

Обозначим как R радиус окружности, описанной около соответствующего правильного N-угольного колеса, а как r — радиус вписанной в него окружности. Выведите в выходной файл число $(R - r)$ с точностью не хуже 10^{-5}.

Примеры

nwheel.in	**nwheel.out**
`3 1`	`0.28867513459481287`
`239 566`	`3.720175440528692`

Задача F. Степень строки

Имя входного файла:	`power.in`
Имя выходного файла:	`power.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Пусть задана строка $s = s_1 s_2 \ldots s_n$. Назовем ее k-ой ($k > 0$) степенью s^k строку $s^k = s_1 s_2 \ldots s_n s_1 s_2 \ldots s_n \ldots \ldots s_1 s_2 \ldots s_n$ (k раз). Например, третьей степенью строки abc является строка $abcabcabc$.

Корнем k степени из строки s называется такая строка t (если она существует), что $t^k = s$.

Ваша задача состоит в том, чтобы написать программу, находящую степень строки или корень из нее.

Формат входного файла

Первая строка входного файла содержит строку s, она содержит только маленькие буквы латинского алфавита и имеет ненулевую длину, не превосходящую 1000.

Вторая строка входного файла содержит целое число $k \neq 0, |k| < 100001$. Если $k > 0$, то необходимо найти k-ую степень строки s, если $k < 0$, то необходимо найти корень степени $|k|$ из s.

Формат выходного файла

В выходной файл выведите строку, являющуюся ответом на задачу. Если длина ответа превосходит 1023 символа, выведите только первые 1023 символа. Если искомой строки не существует — выведите `NO SOLUTION`.

Примеры

power.in	power.out
abc 3	abcabcabc
abcdabcd -2	abcd
abcd -4	NO SOLUTION

Задача G. Поразрядная сортировка

Имя входного файла:	`radix.in`
Имя выходного файла:	`radix.out`
Ограничение по времени:	2 секунды
Ограничение по памяти:	64 мегабайта

Поразрядная сортировка является одним из видов сортировки, которые работают за линейное от размера сортируемого массива время. Такая скорость достигается за счет того, что эта сортировка использует внутреннюю структуру сортируемых объектов.

Изначально этот алгоритм использовался для сортировки перфокарт. Первая его компьютерная реализация была создана в университете MIT Гарольдом Сьюардом (Harold H. Seward).

Опишем алгоритм подробнее. Пусть задан массив строк $s_1, \ldots, s_n$, причем все строки имеют одинаковую длину m. Работа алгоритма состоит из m фаз. На i-ой фазе строки сортируются по i-ой с конца букве.

Происходит это следующим образом. Будем, для простоты, в этой задаче рассматривать строки из цифр от 0 до 9. Для каждой цифры создается «корзина» («bucket»), после чего строки s_i распределяются по «корзинам» в соответствии с i-ой с конца цифрой. Строки, у которых i-ая с конца цифра равна j попадают в j-ую корзину (например, строка 123 на первой фазе попадет в третью корзину, на второй — во вторую, на третьей — в первую). После этого элементы извлекаются из корзин в порядке увеличения номера корзины. Таким образом, после первой фазы строки отсортированы по последней цифре, после двух фаз — по двум последним, ..., после m фаз — по всем.

При важно, чтобы элементы в корзинах сохраняли тот же порядок, что и в исходном массиве (до начала этой фазы). Например, если массив до первой фазы имеет вид: $111, 112, 211, 311$, то элементы по корзинам распределятся следующим образом: в первой корзине будет: $111, 211, 311$, а второй: 112.

Ваша задача состоит в написании программы, детально показывающей работу этого алгоритма на заданном массиве.

Формат входного файла

Первая строка входного файла содержит целое число n ($1 \le n \le 1000$). Последующие n строк содержат каждая по одной строке s_i. Длины всех s_i одинаковы и не превосходят 20. Все s_i состоят только из цифр от 0 до 9.

Формат выходного файла

В выходной файл выведите исходный массив строк s_i, состояние «корзин» после распределения элементов по ним для каждой фазы и отсортированный массив. Следуйте формату, приведенному в примере.

Примеры

radix.in

```
9
12
32
45
67
98
29
61
35
09
```

radix.out

```
Initial array:
12, 32, 45, 67, 98, 29, 61, 35, 09
**********
Phase 1
Bucket 0: empty
Bucket 1: 61
Bucket 2: 12, 32
Bucket 3: empty
Bucket 4: empty
Bucket 5: 45, 35
Bucket 6: empty
Bucket 7: 67
Bucket 8: 98
Bucket 9: 29, 09
**********
Phase 2
Bucket 0: 09
Bucket 1: 12
Bucket 2: 29
Bucket 3: 32, 35
Bucket 4: 45
Bucket 5: empty
Bucket 6: 61, 67
Bucket 7: empty
Bucket 8: empty
Bucket 9: 98
**********
Sorted array:
09, 12, 29, 32, 35, 45, 61, 67, 98
```

Задача H. Различные слова

Имя входного файла:	`words.in`
Имя выходного файла:	`words.out`
Ограничение по времени:	3 секунды
Ограничение по памяти:	128 мегабайт

Дана строка S состоящая из N символов. Назовем ее подстрокой S_{ij} строку с i-го по j-й символ ($i \le j$). Ваша задача — посчитать количество различных подстрок заданной строки.

Формат входного файла

Во входном файле находится одна непустая строка, состоящая из маленьких латинских букв, длиной не более чем 1024 символа.

Формат выходного файла

Выходной файл должен содержать одно число — количество различных подстрок строки S.

Примеры

words.in	**words.out**
abc	6
aaa	3