Задача A. Строки

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Вася посещает занятия по программированию. К сожалению, недавно он заболел и не смог прийти на лекцию. Он смог узнать лишь, что на лекции проходили поиск подстроки в строке. Вася боится, что на следующей тренировке ему придётся искать подстроку в строке, а он до сих пор не знает, как это делается. Помогите ему!

Формат входных данных

В первых двух строчках даны две строки $\mathcal{T}$ и $\mathcal{S}$ ($1 \leqslant |\mathcal{T}| \leqslant 1\,000\,000$, $1 \leqslant |\mathcal{S}| \leqslant 1\,000\,000$). Строки состоят только из маленьких букв «a»–«z» английского алфавита.

Формат выходных данных

Выведите через пробел все начала вхождений строки $\mathcal{S}$ в строку $\mathcal{T}$ в порядке возрастания. Вхождение начинается в позиции k, если $\mathcal{T}_k = \mathcal{S}_1, \mathcal{T}_{k+1} = \mathcal{S}_2, \ldots, \mathcal{T}_{k+|\mathcal{S}|-1} = \mathcal{S}_{|\mathcal{S}|}$. Если вхождений нет, выведите единственное слово «none» вместо списка вхождений.

Примеры

стандартный ввод	*стандартный вывод*
ababcbc abc	3
qwerty asdfgh	none

Задача B. Сравнение строк

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Циклическое расширение $\mathcal{S}^*$ строки $\mathcal{S}$ — это строка $\mathcal{S}$, приписанная сама к себе бесконечное количество раз; к примеру, циклическим расширением строки «bab» является строка «babbabbab...».

Даны длины двух строк $\mathcal{S}$ и $\mathcal{T}$ — числа m и n. Рассмотрим циклические расширения $\mathcal{S}^*$ и $\mathcal{T}^*$ этих строк. Как проверить, равны ли они?

Наивный алгоритм будет проверять строки на равенство, просто сравнивая s_1 с t_1, s_2 с t_2 и так далее. В итоге алгоритм либо найдёт пару несовпадающих символов, либо, если строки равны, будет проводить сравнения бесконечно долго.

Однако понятно, что, если строки не равны, то последняя позиция p, на которой мы можем встретить различие ($s_p \neq t_p$), конечна. Мы хотим узнать, чему равно p, чтобы улучшить алгоритм сравнения так: новый алгоритм будет сравнивать символ s_1 с t_1, s_2 с t_2 и так далее, пока либо не найдёт несовпадение $s_q \neq t_q$ на позиции $q \leqslant p$, либо, сравнив первые p пар и обнаружив соответствие символов в каждой паре, не докажет тем самым равенство строк $\mathcal{S}^*$ и $\mathcal{T}^*$.

Для данных длин строк m и n приведите пример строк $\mathcal{S}$ и $\mathcal{T}$ соответствующей длины, циклические расширения которых различны, но первое несовпадение встречается как можно позже.

Формат входных данных

В первой строке ввода заданы два числа через пробел — это числа m и n ($1 \leqslant m, n \leqslant 100$).

Формат выходных данных

Выведите в первой строке строку $\mathcal{S}$ из m символов, а во второй — строку $\mathcal{T}$ из n символов. Строки могут содержать только маленькие буквы английского алфавита ('a'–'z').

Пример

стандартный ввод	*стандартный вывод*
2 4	aa aaab

Пояснение к примеру

Для $m = 2$ и $n = 4$ значение p равно четырём, и оно достигается на строках «aa» и «aaab», циклические расширения которых — строки «aaaa...» и «aaab...» — различаются в четвёртом символе.

Задача C. Двухкратная подстрока

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Дана строка S длины n и число k. Найдите в строке S такую подстроку длины k, которая встречается в ней по крайней мере два раза, или выясните, что такой подстроки нет.

Формат входных данных

В первой строке задана строка S; её длина n — от 1 до 100 000 символов, включительно. Во второй строке задано целое число k ($1 \leqslant k \leqslant n$). Строка состоит только из маленьких букв английского алфавита.

Формат выходных данных

Если подстроки длины k, встречающейся хотя бы два раза, не существует, выведите слово «NONE». В противном случае выведите любую из таких подстрок.

Примеры

стандартный ввод	*стандартный вывод*
ast 1	NONE
blinkingblueblogger 2	in
aaaaaab 5	aaaaa

Задача D. Сравнения подстрок

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Дана строка. Нужно уметь отвечать на запросы следующего вида: равны ли подстроки $[a..b]$ и $[c..d]$.

Формат входных данных

Сперва строка S (не более 10^5 строчных латинских букв). Далее число M — количество запросов.

В следующих M строках запросы, каждый — в виде четырёх целых чисел a, b, c, d ($0 \leqslant m \leqslant 10^5$, $1 \leqslant a \leqslant b \leqslant |S|$, $1 \leqslant c \leqslant d \leqslant |S|$).

Формат выходных данных

Выведите M строк: на каждый запрос выведите «Yes», если подстроки совпадают, и «No» иначе.

Пример

стандартный ввод	*стандартный вывод*
trololo 3 1 7 1 7 3 5 5 7 1 1 1 5	Yes Yes No

Задача E. Плохое хеширование

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Маленький мальчик Петя решает задачу про строки. Не будем останавливаться подробно на её условии. Для нас важно только то, что задача сводится к следующей: дано несколько строк, состоящих исключительно из маленьких букв английского алфавита, и нужно брать пары строк и проверять, равны ли они.

Петя придумал, как делать это быстро: сначала он посчитал *хеш-функцию* от каждой строки, а затем, когда нужно сравнить две строки, просто сравнивает значения хеш-функции от этих строк. Конечно, если значения хеш-функции различны, строки тоже различны. А вот если значения одинаковы, это ещё не гарантирует равенства самих строк.

Мы хотим *взломать* Петино решение, то есть придумать такие две различные строки, что значения хеш-функции от них одинаковы. Чтобы сделать это, разберёмся, что за функцию Петя использует для хеширования.

При ближайшем рассмотрении оказалось, что Петя реализовал *полиномиальный хеш* от строки. Полиномиальный хеш задаётся множителем p и модулем q. Для пустой строки ε значение хеш-функции $h(\varepsilon) = 0$, а для любой строки S и любого символа c хеш-функция рекуррентно определяется как $h(S + c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Здесь $\text{code}(c)$ — это ASCII-код символа c. Как известно, коды маленьких букв английского алфавита идут подряд: $\text{code}(\texttt{'a'}) = 97$, $\text{code}(\texttt{'b'}) = 98$, …, $\text{code}(\texttt{'z'}) = 122$. Можно выписать и нерекуррентную формулу: если строка $S = s_1 s_2 \dots s_n$, то $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

Это достаточно распространённый метод хеширования, однако Петя не подумал о том, что его решение могут взломать, и допустил две существенные ошибки при выборе p и q. Во-первых, модуль q слишком мал, он равен всего лишь 2^{32} (Петя просто считает значение хеш-функции в 32-битном целом беззнаковом типе данных и не обращает внимания на переполнение). Во-вторых, при этом множитель p чётный.

Зная множитель p, взломайте решение Пети.

Формат входных данных

Первая строка ввода содержит целое число p — множитель полиномиального хеширования ($0 < p < 2^{32}$). Гарантируется, что p чётно.

Формат выходных данных

В первых двух строках выведите две различные строки S и T, для которых $h(S) = h(T)$. Строки должны состоять исключительно из маленьких букв английского алфавита (ASCII-коды 97–122) и иметь длину от 1 до 100 000 символов. Заметим, что длины строк не обязательно должны совпадать. Если возможных ответов несколько, разрешается вывести любой из них.

Примеры

стандартный ввод	*стандартный вывод*
4	ae ba
1000	vabavydw budqhmng

Пояснения к примерам

В первом примере $h(S) = (97 \cdot 4 + 101) \bmod 2^{32} = 489$ и $h(T) = (98 \cdot 4 + 97) \bmod 2^{32} = 489$.

Во втором примере

$$h(S) = 118\,097\,098\,097\,118\,121\,100\,119 \bmod 2^{32} = 834\,470\,743 \text{ и}$$
$$h(T) = 98\,117\,100\,113\,104\,109\,110\,103 \bmod 2^{32} = 834\,470\,743.$$

Задача F. Плохое хеширование 2

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Маленький мальчик Петя решает задачу про строки. Не будем останавливаться подробно на её условии. Для нас важно только то, что задача сводится к следующей: дано несколько строк, состоящих исключительно из маленьких букв английского алфавита, и нужно брать пары строк и проверять, равны ли они.

Петя придумал, как делать это быстро: сначала он посчитал *хеш-функцию* от каждой строки, а затем, когда нужно сравнить две строки, просто сравнивает значения хеш-функции от этих строк. Конечно, если значения хеш-функции различны, строки тоже различны. А вот если значения одинаковы, это ещё не гарантирует равенства самих строк.

Мы хотим *взломать* Петино решение, то есть придумать такие две различные строки, что значения хеш-функции от них одинаковы. Чтобы сделать это, разберёмся, что за функцию Петя использует для хеширования.

При ближайшем рассмотрении оказалось, что Петя реализовал *полиномиальный хеш* от строки. Полиномиальный хеш задаётся множителем p и модулем q. Для пустой строки ε значение хеш-функции $h(\varepsilon) = 0$, а для любой строки S и любого символа c хеш-функция рекуррентно определяется как $h(S + c) = (h(S) \cdot p + \text{code}(c)) \bmod q$. Здесь $\text{code}(c)$ — это ASCII-код символа c. Как известно, коды маленьких букв английского алфавита идут подряд: $\text{code}(\texttt{'a'}) = 97$, $\text{code}(\texttt{'b'}) = 98$, …, $\text{code}(\texttt{'z'}) = 122$. Можно выписать и нерекуррентную формулу: если строка $S = s_1 s_2 \dots s_n$, то $h(S) = (\text{code}(s_1) \cdot p^{n-1} + \text{code}(s_2) \cdot p^{n-2} + \dots + \text{code}(s_n) \cdot p^0) \bmod q$.

Это достаточно распространённый метод хеширования, однако Петя не подумал о том, что его решение могут взломать, и допустил две существенные ошибки при выборе p и q. Во-первых, модуль q слишком мал, он равен всего лишь 2^{32} (Петя просто считает значение хеш-функции в 32-битном целом беззнаковом типе данных и не обращает внимания на переполнение). Во-вторых, при этом множитель p нечётный.

Зная множитель p, взломайте решение Пети.

Формат входных данных

Первая строка ввода содержит целое число p — множитель полиномиального хеширования ($0 < p < 2^{32}$). Гарантируется, что p нечётно.

Формат выходных данных

В первых двух строках выведите две различные строки S и T, для которых $h(S) = h(T)$. Строки должны состоять исключительно из маленьких букв английского алфавита (ASCII-коды 97–122) и иметь длину от 1 до 100 000 символов. Заметим, что длины строк не обязательно должны совпадать. Если возможных ответов несколько, разрешается вывести любой из них.

Пример

стандартный ввод	*стандартный вывод*
3	ad ba

Пояснение к примеру

В примере $h(S) = (97 \cdot 3 + 100) \bmod 2^{32} = 391$ и $h(T) = (98 \cdot 3 + 97) \bmod 2^{32} = 391$.

Задача G. Циклический сдвиг или нет

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Это задача с двойным запуском.

Дана строка s из n двоичных цифр. Также дана ещё одна строка t, содержащая n двоичных цифр. Выясните, является ли одна строка циклическим сдвигом другой.

Строка s является циклическим сдвигом t, если существует число k ($0 \leqslant k < n$), для которого $s_1 s_2 \dots s_n = t_{k+1} t_{k+2} t_n t_1 t_2 \dots t_k$.

В чём сложность? — спросите вы. Ваше решение запускается два раза, и строки даны в разных запусках.

Первый запуск

При первом запуске решение получает строку s. В первой строке записано слово «`first`». Во второй строке задано целое число n — длина строки ($1 \leqslant n \leqslant 100\,000$). В третьей строке задана сама строка s, состоящая из n двоичных цифр без пробелов.

Выведите подсказку h — строку, содержащую от 1 до 100 произвольных символов с ASCII-кодами от 32 до 126 включительно.

Второй запуск

При втором запуске решение получает строку t и подсказку h. В первой строке записано слово «`second`». Во второй строке задано целое число n — длина строки (та же, что при первом запуске). В третьей строке задана сама строка t, состоящая из n двоичных цифр без пробелов. Наконец, в четвёртой строке дана подсказка h, выведенная при первом запуске.

Выведите «`YES`», если t является циклическим сдвигом s, и «`NO`» в противном случае. Каждую букву можно выводить в любом регистре.

В каждом тесте строки s и t зафиксированы заранее.

Пример

В каждом тесте входные данные при втором запуске зависят от того, что вывело решение при первом запуске.

Далее показаны два запуска какого-то решения на первом тесте.

стандартный ввод	*стандартный вывод*

`first`
`6`
`011001`
	`#@! 001011 !@#`

`second`
`6`
`100101`
`#@! 001011 !@#`
	`YES`

Задача H. Период строки

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Строка S имеет период T, если

$$\exists n > 0 : S = T^n = \underbrace{TT\ldots T}_{n}.$$

Вам дана строка S. Ваша задача — найти минимальную по длине строку T, для которой $S = T^n$ при некотором $n \in \mathbb{N}$.

Формат входных данных

Строка S длиной от 1 до 10^6 символов.

Формат выходных данных

Единственное число — длина T.

Пример

стандартный ввод	*стандартный вывод*
abaabaabaabaaba	3

Задача I. Два генератора

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Строка p называется *генератором* строки s, если s совпадает с каким-то префиксом p^*, и p — кратчайшая из таких строк. Здесь p^* — это строка p, приписанная к себе самой бесконечное количество раз.

Рассмотрим строку t. Представим её в виде конкатенации $t = a+b$ (строки a и b могут быть пустыми). После этого найдём генератор строки a и генератор строки b. Найдите такое разбиение t на a и b, что суммарная длина этих двух генераторов минимальна, и выведите эту длину.

Формат входных данных

В первой строке задано целое число n — длина строки t ($1 \leqslant n \leqslant 300\,000$). Во второй строке записана сама строка t, состоящая из строчных букв английского алфавита.

Формат выходных данных

Выведите одно целое число: минимально возможную суммарную длину двух генераторов.

Примеры

стандартный ввод	*стандартный вывод*
10 abcabcaddd	4
3 aaa	1

Задача J. От префикс-функции к z-функции

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Префикс-функция $p(i)$ для строки $s = s_1 s_2 \dots s_n$ определяется от позиции i ($1 \leqslant i \leqslant n$) в строке так: $p(i)$ — это максимальная длина собственного префикса строки $s_1 s_2 \dots s_i$, равного её собственному суффиксу. Напомним, что *собственный префикс* строки $s = s_1 s_2 \dots s_n$ — это строка $s_1 s_2 \dots s_r$ для некоторого $r < n$. Аналогично, *собственный суффикс* строки $s = s_1 s_2 \dots s_n$ — это строка $s_l s_2 \dots s_n$ для некоторого $l > 1$.

Z-функция $z(i)$ для строки $s = s_1 s_2 \dots s_n$ определяется от позиции i ($1 \leqslant i \leqslant n$) в строке так: $z(1) = 0$, а для $i > 1$ значение $z(i)$ — это максимальное число, для которого строки $s_1 s_2 \dots s_{z(i)}$ и $s_i s_{i+1} \dots s_{i+z(i)-1}$ совпадают.

Даны длина строки n и значения префикс-функции $p(1)$, $p(2)$, …, $p(n)$ для этой строки. Найдите для этой строки значения z-функции $z(1)$, $z(2)$, …, $z(n)$.

Формат входных данных

В первой строчке задано целое число n ($1 \leqslant n \leqslant 1\,000\,000$). Во второй строчке заданы n чисел через пробел — значения префикс-функции $p(1)$, $p(2)$, …, $p(n)$. Гарантируется, что существует строка длины n, состоящая из маленьких букв английского алфавита, для которой префикс-функция от позиций 1, 2, …, n принимает данные значения.

Формат выходных данных

В первой строчке выведите n чисел через пробел — значения z-функции для строки, имеющей данную префикс-функцию.

Примеры

стандартный ввод	*стандартный вывод*
6 0 0 1 2 3 4	0 0 4 0 2 0
7 0 0 0 1 2 3 4	0 0 0 4 0 0 1
4 0 0 0 0	0 0 0 0

Задача K. От z-функции к префикс-функции

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Z-функция $z(i)$ для строки $s = s_1 s_2 \dots s_n$ определяется от позиции i ($1 \leqslant i \leqslant n$) в строке так: $z(1) = 0$, а для $i > 1$ значение $z(i)$ — это максимальное число, для которого строки $s_1 s_2 \dots s_{z(i)}$ и $s_i s_{i+1} \dots s_{i+z(i)-1}$ совпадают.

Префикс-функция $p(i)$ для строки $s = s_1 s_2 \dots s_n$ определяется от позиции i ($1 \leqslant i \leqslant n$) в строке так: $p(i)$ — это максимальная длина собственного префикса строки $s_1 s_2 \dots s_i$, равного её собственному суффиксу. Напомним, что *собственный префикс* строки $s = s_1 s_2 \dots s_n$ — это строка $s_1 s_2 \dots s_r$ для некоторого $r < n$. Аналогично, *собственный суффикс* строки $s = s_1 s_2 \dots s_n$ — это строка $s_l s_2 \dots s_n$ для некоторого $l > 1$.

Даны длина строки n и значения z-функции $z(1)$, $z(2)$, …, $z(n)$ для этой строки. Найдите для этой строки значения префикс-функции $p(1)$, $p(2)$, …, $p(n)$.

Формат входных данных

В первой строчке задано целое число n ($1 \leqslant n \leqslant 1\,000\,000$). Во второй строчке заданы n чисел через пробел — значения z-функции $z(1)$, $z(2)$, …, $z(n)$. Гарантируется, что существует строка длины n, состоящая из маленьких букв английского алфавита, для которой z-функция от позиций 1, 2, …, n принимает данные значения.

Формат выходных данных

В первой строчке выведите n чисел через пробел — значения префикс-функции для строки, имеющей данную z-функцию.

Примеры

стандартный ввод	*стандартный вывод*
6 0 0 4 0 2 0	0 0 1 2 3 4
7 0 0 0 4 0 0 1	0 0 0 1 2 3 4
4 0 0 0 0	0 0 0 0

Задача L. Ретростока

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Строкой S называется последовательность символов $S_1S_2\dots S_n$, где $|S| = n$ — это *длина* строки S.

Для любого k ($1 \leqslant k \leqslant |S|$) *$k$-м префиксом* строки S называется строка $S_1S_2\dots S_k$ длины k. Если $k < |S|$, то префикс называется *собственным*.

Аналогично для любого k ($1 \leqslant k \leqslant |S|$) *$k$-м суффиксом* строки S называется строка $S_{|S|-k+1}S_{|S|-k+2}\dots S_{|S|}$ длины k. Если $k < |S|$, то суффикс также называется *собственным*.

Назовём *числом повторяемости* строки S количество её различных собственных суффиксов, каждый из которых совпадает с префиксом той же длины, что и этот суффикс.

Назовём строку *ретростокой*, если её число повторяемости строго больше чисел повторяемости всех её собственных префиксов.

Дана строка S. Нужно найти её префикс максимальной длины (не обязательно собственный), являющийся ретростокой.

Формат входных данных

В первой строке записана строка S ($1 \leqslant |S| \leqslant 1\,000\,000$). Строка содержит лишь символы с ASCII-кодами от 33 до 126.

Формат выходных данных

В первой строке должен быть выведен префикс S максимальной длины, являющийся ретростокой.

Примеры

стандартный ввод	*стандартный вывод*
z	z
aabaabaaabaabaaabaaba	aabaabaaabaabaa

Задача M. Суффиксы

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Назовём *строкой* последовательность из маленьких букв английского алфавита. Строкой, например, является пустая последовательность «», слово «`aabaf`» или бесконечная последовательность букв «`a`».

Определим *i-й суффикс* S_i строки S: это просто строка S, из которой вырезаны первые i букв. Так, для строки $S =$ «`aabaf`» суффиксы будут такими:

$$\begin{aligned} S_0 &= \text{«aabaf»} \\ S_1 &= \text{«abaf»} \\ S_2 &= \text{«baf»} \\ S_3 &= \text{«af»} \\ S_4 &= \text{«f»} \\ S_5 = S_6 = S_7 = \ldots &= \text{«»} \end{aligned}$$

Суффиксы определены для всех $i \geqslant 0$.

Циклическое расширение $S^\star$ конечной строки S — это строка, полученная приписыванием её к самой себе бесконечное количество раз. Так,

$$\begin{aligned} S^\star = S_0^\star &= \text{«aabafaabafaa...»} \\ S_1^\star &= \text{«abafabafabaf...»} \\ S_2^\star &= \text{«bafbafbafbaf...»} \\ S_3^\star &= \text{«afafafafafaf...»} \\ S_4^\star &= \text{«fffffffffff...»} \\ S_5^\star = S_6^\star = S_7^\star = \ldots &= \text{«»} \end{aligned}$$

По данной строке S выясните, сколько её суффиксов S_i имеют такое же циклическое расширение, как и сама строка S, то есть количество таких i, что $S^\star = S_i^\star$.

Формат входных данных

В первой и единственной строке задана строка S, состоящая из не менее чем одной и не более чем $100\,000$ маленьких английских букв «`a-z`».

Формат выходных данных

Выведите одно число — количество суффиксов строки S, имеющих такое же циклическое расширение, как и она сама.

Примеры

стандартный ввод	*стандартный вывод*
`aa`	`2`
`ab`	`1`
`qqqq`	`4`
`xyzzyxy`	`1`

Задача N. Минимальный суффикс

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Как известно, *k-м суффиксом* строки $S = s_1 s_2 \dots s_n$ называется строка $S_k = s_k s_{k+1} s_{k+2} \dots s_n$.

Например, для строки $S =$ `abaca` суффиксы будут такие: $S_1 =$ `abaca`, $S_2 =$ `baca`, $S_3 =$ `aca`, $S_4 =$ `ca`, $S_5 =$ `a`, а все последующие суффиксы пусты.

В этой задаче требуется найти лексикографически минимальный непустой суффикс заданной строки S.

Формат входных данных

В первой строке записана строка S, состоящая только из маленьких букв английского алфавита. Длина этой строки — от 1 до 100 000 букв, включительно.

Формат выходных данных

В первой строке выведите суффикс S_k, являющийся лексикографически минимальным непустым суффиксом строки S.

Примеры

стандартный ввод	*стандартный вывод*
abcde	abcde
abaca	a
bcdeabc	abc

Задача O. Палиндромы

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Строка называется палиндромом, если она одинаково читается как слева направо, так и справа налево. Например, `abba` — палиндром, а `omax` — нет. Для строки S будем обозначать $S[i..j]$ её подстроку длины $j - i + 1$ с i-й по j-ю позицию включительно (позиции нумеруются с единицы).

Для заданной строки S длины N требуется найти количество таких пар (i, j), что $1 \leqslant i < j \leqslant n$ и подстрока $S[i..j]$ является палиндромом.

Формат входных данных

Ввод содержит одну строку S длины N, состоящую из маленьких английских букв ($1 \leqslant N \leqslant 100\,000$).

Формат выходных данных

Выведите искомое количество пар.

Примеры

стандартный ввод	*стандартный вывод*
aaa	3
abba	2
omax	0

Задача P. Разбиение на палиндромы

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	1 секунда
Ограничение по памяти:	512 мебибайт

Дана строка, состоящая из символов с ASCII-кодами от 32 до 126, включительно. Можно ли разбить эту строку на палиндромы чётной длины?

Формат входных данных

Первая строка ввода содержит число n — количество тестов ($1 \leqslant n \leqslant 100$). Следующие n строк содержат сами тесты. Длина каждой строки не превышает 10^6 символов. Общий объём ввода не превышает трёх мегабайт.

Формат выходных данных

Выведите ответ на каждый тест на отдельной строке. Выводите «`YES`», если строку можно разбить на палиндромы чётной длины, и «`NO`» в противном случае.

Пример

стандартный ввод	*стандартный вывод*
3 madam aA aabb	NO NO YES