Problem A. НВП

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Дана последовательность, требуется найти длину её наибольшей возрастающей подпоследовательности. Подпоследовательностью последовательности называется некоторый набор её элементов, не обязательно стоящих подряд.

Input

В первой строке входных данных задано число N — длина последовательности ($1 \le N \le 1000$). Во второй строке задается сама последовательность (разделитель — пробел). Элементы последовательности — целые числа, не превосходящие 10000 по абсолютной величине.

Output

Требуется вывести длину наибольшей строго возрастающей подпоследовательности.

Examples

standard input	standard output
6 3 29 5 5 28 6	3
10 4 8 2 6 2 10 6 29 58 9	5

Problem B. Наибольшая общая подпоследовательность

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second(s)
Memory limit:	512 MiB

Общей подпоследовательностью двух строк s_1 и s_2 называется пара последовательностей индексов $(\{a_i\}, \{b_i\})$ такая, что $a_1 < a_2 < \ldots < a_k$, $b_1 < b_2 < \ldots < b_k$, and $s_1[a_i] = s_2[b_i]$ for all $1 \le i \le k$.

Найдите наибольшую общую подпоследовательность двух строк.

Input

Первая и вторая строки входа содержат две непустые строки, каждая из которых состоит из строчных латинских букв. Длина каждой строки не превосходит 100.

Output

В первой строке выведите целое число k — длину наибольшей общей подпоследовательности. Во второй выведите k целых чисел — индексы символов наибольшей общей подпоследовательности в первой строке, отсортированные по возрастанию. В третьей, аналогично — отсортированные по возрастанию индексы символов наибольшей общей подпоследовательности во второй строке. Символы в строках занумерованы с 1.

Если способов выбрать наибольшую общую подпоследовательность несколько, выведите любой из них.

Example

standard input	standard output
abcd cxbydz	2 3 4 1 5

Problem C. Шаблоны

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 MiB

Многие операционные системы используют шаблоны для ссылки на группы объектов: файлов, пользователей, и т. д. Ваша задача — реализовать простейший алгоритм проверки шаблонов для имен файлов.

В этой задаче алфавит состоит из маленьких букв английского алфавита и точки (‘.’). Шаблоны могут содержать произвольные символы алфавита, а также два специальных символа: ‘?’ и ‘*’. Знак вопроса (‘?’) соответствует ровно одному произвольному символу. Звездочка ‘*’ соответствует подстроке произвольной длины (возможно, нулевой). Символы алфавита, встречающиеся в шаблоне, отображаются на ровно один такой же символ в проверяемой строчке. Строка считается подходящей под шаблон, если символы шаблона можно последовательно отобразить на символы строки таким образом, как описано выше. Например, строчки “ab”, “aab” и “beda.” подходят под шаблон “*a?”, а строчки “bebe”, “a” и “ba” — нет.

Input

Первая строка входного файла определяет шаблон P. Вторая строка S состоит только из символов алфавита. Ее необходимо проверить на соответствие шаблону. Длины обеих строк не превосходят 10 000. Строки могут быть пустыми — будьте внимательны!

Output

Если данная строка подходит под шаблон, выведите YES. Иначе выведите NO.

Examples

standard input	standard output
k?t*n kitten	YES
k?t?n kitten	NO

Problem D. Редакционное расстояние

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 MiB

В информатике *редакционным расстоянием* между двумя строками называется минимальное количество добавлений, удалений и замен символов, при помощи которых можно из одной строки получить другую. К примеру, редакционное расстояние между строками “ab” и “ab” равно нулю, так как строки равны между собой безо всяких изменений; расстояние между строками “short” и “ports” равно трём: в слове “short” нужно удалить из начала букву ‘s’, заменить ‘h’ на ‘p’ и добавить в конец букву ‘s’. Редакционное расстояние также называют *расстоянием Левенштейна*.

Найдите редакционное расстояние между двумя заданными строками.

Input

В первой строчке входного файла задана одна строка, во второй "— другая.

Длины строк от 1 до 100.

Output

В выходной файл выведите единственное число "— редакционное расстояние между двумя заданными строками.

Examples

standard input	standard output
ab ab	0
short ports	3

Problem E. Наибольшая пилообразная подпоследовательность

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Числовая последовательность называется пилообразной если каждый ее член (кроме первого и последнего) либо больше обоих своих соседей, либо меньше обоих соседей. Например, последовательность 1, 2, 1, 3, 2 является пилообразной, а 1, 2, 3, 1, 2 — нет, поскольку $1 < 2 < 3$. Любая последовательность из одного элемента является пилообразной. Последовательность из двух элементов является пилообразной, если ее элементы не равны.

Дана последовательность. Требуется определить, какое наименьшее количество ее членов нужно вычеркнуть, чтобы оставшаяся последовательность оказалась пилообразной.

Input

В первой строке входного файла записано одно число N ($1 \le N \le 10^5$) — количество членов последовательности. Во второй строке записано N натуральных чисел, не превосходящих 10^4 — члены последовательности.

Output

В выходной файл выведите одно число — минимальное количество членов, которые необходимо вычеркнуть.

Examples

standard input	standard output
5 1 2 3 1 2	1
5 1 2 1 3 2	0

Problem F. Гангстеры

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

N гангстеров собираются в ресторан. i-й гангстер приходит в момент времени T_i и имеет богатство P_i.

Дверь ресторана имеет $K+1$ степень открытости, они обозначаются целыми числами из интервала [0, K]. Степень открытости двери может изменяться на единицу в единицу времени, то есть дверь может открыться на единицу, закрыться на единицу или остаться в том же состоянии. В начальный момент времени дверь закрыта (степень открытости 0).

i-й гангстер заходит в ресторан, только если дверь открыта специально для него, то есть когда степень открытости двери соответствует его полноте S_i. Если в момент, когда гангстер подходит к ресторану, степень открытости двери не соответствует его полноте, он уходит и больше не возвращается. Ресторан работает в интервале времени [0, T].

Требуется собрать гангстеров с максимальным суммарным богатством в ресторане, открывая и закрывая дверь соответствующим образом.

Input

В первой строке находятся числа N, K, T, во второй - $T_1, T_2, ..., T_N$, в третьей - $P_1, P_2, ..., P_N$. в четвёртой - $S_1, S_2, ..., S_N$. Числа в строках разделены пробелами.

$1 \le N \le 100$, $1 \le K \le 100$, $1 \le T \le 3 \cdot 10^4$, $0 \le T_i \le T$, $1 \le P_i \le 300$, $1 \le S_i \le K$, все числа целые.

Output

Вывести одно число - максимальное суммарное богатство гангстеров, попавших в ресторан. Если зайти не удалось никому, вывести 0.

Examples

standard input	standard output
2 10 20 10 16 10 11 10 7	21
1 10 20 15 10 10	10

Problem G. Упорядочьте шарики

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second(s)
Memory limit:	512 MiB

При игре в новую игру (некоторый гибрид боулинга и бильярда) используется N шариков, пронумерованных числами от 1 до N. В начале игры эти шарики должны быть выложены в линию в порядке своих номеров. В процессе игры их порядок может меняться.

Для того, чтобы упорядочить шарики перед началом следующей партии, используется следующее устройство. Это устройство состоит из головки, которая, перемещаясь над шариками, может «засасывать» и «выплевывать» шарики. Чтобы получить большее представление об этом устройстве, представьте себе пылесос, который может засасывать шарики, перемешаться в нужное место, и там, включаясь на продув в обратном направлении, шарики «выплевывать».

При засасывании шарика все шарики, которые находились правее засасываемого, сдвигаются влево. «Выплюнуть» шарик можно между любыми двумя шариками (а также перед первым шариком или после последнего), тогда выплевываемый шарик вставляется между этими шариками, и все шарики, которые находятся правее вставляемого, сдвигаются вправо.

В устройство может быть одновременно засосано больше одного шарика, при этом при выплевывании шарика первым выплевывается последний засосанный шарик, затем - предпоследний и т.д. (т.е. устройство работает по принципу стека). Шарики выплевываются по одному, т.е. можно выплюнуть только один шарик, остальные оставив внутри устройства (при этом дальше можно как продолжать «выплевывать» шарики в том же или в другом месте, так и засасывать новые шарики).

Наиболее энергоемкой из описанных операций является операция засасывания шарика, поэтому хочется минимизировать количество именно таких операций.

Напишите программу, которая по данному начальному расположению шариков определит минимальное количество операций засасывания, которое нужно, чтобы расположить шарики в порядке их номеров.

Input

Во входном файле задано сначала число N — количество шариков ($1 \le N \le 1000$). Далее идет N чисел, задающих номера шариков в порядке слева направо в их текущем расположении (каждое число — от 1 до N, и каждое из чисел встречается в последовательности ровно один раз).

Output

В выходной файл выведите одно число — минимальное количество операций засасывания шарика, которое потребуется, чтобы расположить шарики в порядке их номеров.

Examples

standard input	standard output
3 2 1 3	1
4 4 3 2 1	3