1. Сыщик

ограничение по времени на тест

2 seconds

ограничение по памяти на тест

256 megabytes

ввод

стандартный ввод

вывод

стандартный вывод

Вася играет с друзьями в Сыщика. Правила этой игры таковы: тот, кто играет в первый раз, то есть Вася, — сыщик, он должен расследовать «преступление» и выяснить, что же происходит. Он может задавать абсолютно любые вопросы, на которые можно ответить «Да» или «Нет». Все остальные заранее сговариваются отвечать на все вопросы следующим образом: если последняя буква вопроса — гласная, они отвечают «Да», а если последняя буква — согласная, они отвечают «Нет». Сыщик об этом естественно не знает, и его цель — понять это.

К сожалению, Вася не отличается сообразительностью. После 5 часов непрерывных глупых вопросов всем кроме Васи надоело. Поэтому Васины друзья просят вас написать программу, которая отвечала бы на вопросы за них.

Гласные буквы английского алфавита: A, E, I, O, U, Y

Согласные буквы английского алфавита: B, C, D, F, G, H, J, K, L, M, N, P, Q, R, S, T, V, W, X, Z

**Входные данные**

В единственной строке записан вопрос — непустая строка из больших и маленьких букв латинского алфавита, пробелов и одного знака вопроса. Длина строки не превосходит 100. Гарантируется, что знак вопроса встречается в строке ровно один раз — в качестве последнего символа, а строка содержит хотя бы одну букву.

**Выходные данные**

Выведите ответ на вопрос: YES в случае ответа «Да», NO в случае ответа «Нет».

Помните, что при ответе на вопрос учитывается последняя **буква**, а не последний символ. Т. е. пробелы и знак вопроса за буквы не считаются.

**Примеры**

**входные данные**

Is it a melon?

**выходные данные**

NO

**входные данные**

Is it an apple?

**выходные данные**

YES

**входные данные**

Is it a banana ?

**выходные данные**

YES

**входные данные**

Is it an apple and a banana simultaneouSLY?

**выходные данные**

YES

2. Сумма

ограничение по времени на тест

2 seconds

ограничение по памяти на тест

256 megabytes

ввод

стандартный ввод

вывод

стандартный вывод

Вася изучает позиционные системы счисления. К сожалению, он часто забывает записать основание системы счисления, в которой записано выражение. Однажды он увидел в своей тетради запись *a* + *b* = ?, при этом основание системы счисления нигде записано не было. Теперь Вася должен выбрать некоторое основание *p* и считать что выражение записано в *p*-ичной системе счисления. Вася понял, что при разных основаниях он может получать разные ответы, а некоторые основания даже недопустимы. Например, выражение 78 + 87 в 16-ичной системе счисления равняется *FF*16, в 15-ичной: 11015, в 10-ичной: 16510, в 9-ичной: 1769, а в системах счисления с основанием 8 или меньше данное выражение недопустимо, т. к. все цифры должны быть строго меньше основания системы счисления. Васе стало интересно, какова длина самого длинного возможного значения выражения. Помогите ему найти эту длину.

Под длиной числа следует понимать количество цифр в нем. Например, длина самого длинного ответа для 78 + 87 = ? — это 3. Это получается, например, в 15-ичной (11015), 10-ичной (16510), 9-ичной (1769) и некоторых других системах счисления.

**Входные данные**

В первой строке через пробел записано два числа *a* и *b* (1 ≤ *a*, *b* ≤ 1000) — заданные слагаемые.

**Выходные данные**

Выведите одно число — длину наидлиннейшего ответа.

**Примеры**

**входные данные**

78 87

**выходные данные**

3

**входные данные**

1 1

**выходные данные**

2

# 3. Две Строки

ограничение по времени на тест

1 seconds

ограничение по памяти на тест

256 megabytes

ввод

стандартный ввод

вывод

стандартный вывод

Циклическим сдвигом строки s1s2…sn на k позиций назовем строку sk+1sk+2…sns1..sk. Например, циклическим сдвигом строки «abcde» на две позиции является строка «cdeab». В этой задаче далее будут рассматриваться только строки, состоящие из десятичных цифр от 0 до 9. Произвольной такой строке, первый символ которой не является нулем, можно сопоставить число, десятичной записью которого она является. Строкам, которые начинаются с нуля, никакое число сопоставляться не будет. Например, строке 123 сопоставляется число сто двадцать три, а строке 0123 не сопоставляется никакое число.

Пусть заданы две строки: s и t. Обозначим как S набор всех циклических сдвигов строки s, а как T – набор всех циклических сдвигов строки t. Например, если s = «1234», то S содержит строки «1234», «2341», «3412», «4123». Обозначим также как NUM(A) набор чисел, соответствующих строкам из набора A.

Требуется написать программу, которая по строкам s и t определит, максимальное число, представимое в виде разности (x – y), где x принадлежит NUM(S), а y принадлежит NUM(T). Например, если s = «25», t = «12», то NUM(S) содержит числа 25 и 52, NUM(T) – числа 12 и 21; их попарными разностями будут: 25 – 12 = 13, 25 – 21 = 4, 52 – 12 = 40, 52 – 21 = 31. Из этих разностей максимальным числом является 40.

## Входные данные

Первая строка входного файла INPUT.TXT содержит строку s, вторая строка входного файла – строку t. Обе строки непустые. Они содержат только цифры, из которых хотя бы одна не является нулем. Строки имеют длину не более 3000 символов.

## Выходные данные

В выходной файл OUTPUT.TXT выведите искомое число без ведущих нулей.

**Примеры**

**входные данные**

25

12

**выходные данные**

40

**входные данные**

100

1

**выходные данные**

99

4. Need For Brake

ограничение по времени на тест

4 seconds

ограничение по памяти на тест

256 megabytes

ввод

стандартный ввод

вывод

стандартный вывод

Вася играет в Need For Brake. Играет потому, что ему подарили на день рождения новенький компьютерный руль! Теперь то он уж точно займет первое место в чемпионате в его любимой компьютерной игре в жанре авто-симуляторов!

В чемпионате, состоящем из некоторого количества заездов, принимают участие *n* гонщиков. По результатам каждого заезда гонщики занимают места с 1 по *n*-ое (никакие два гонщика не разделяют одно место) и первые *m* мест являются призовыми. За *i*-ое призовое место гонщик получает *bi* очков, которые прибавляются к его очкам, заработанным за все предыдущие заезды. Известно, что к данному моменту гонщик *i* набрал *ai* очков. По итогам чемпионата гонщики сортируются в порядке убывания суммы очков. При равном количестве очков сортировка производится по возрастанию имени гонщика в лексикографическом порядке.

К сожалению, чемпионат уже подошел к концу и остался всего лишь один заезд. Вася решил узнать какое самое высокое и самое низкое место он может занять по итогам чемпионата.

**Входные данные**

В первой строке задано число *n* — количество гонщиков (1 ≤ *n* ≤ 105). В каждой из следующих *n* строк через пробел записаны *si* и *ai* — псевдоним, под которым выступает гонщик (непустая строка, состоящая из не более чем 20 маленьких латинских букв), и количество очков этого гонщика (0 ≤ *ai* ≤ 106). Гонщики заданы в произвольном порядке.

В следующей строке находится число *m* (0 ≤ *m* ≤ *n*). Далее идет *m* неотрицательных целых чисел *bi*. *i*-ое число в этой строке равно числу бонусных очков за *i*-ое призовое место (0 ≤ *bi* ≤ 106).

В последней строчке задан псевдоним, под которым выступает Вася.

**Выходные данные**

Вывести два числа — самое высокое и самое низкое места, которые может занять Вася по итогам чемпионата.

**Примеры**

**входные данные**

3  
teama 10  
teamb 20  
teamc 40  
2  
10 20  
teama

**выходные данные**

2 3

**входные данные**

2  
teama 10  
teamb 10  
2  
10 10  
teamb

**выходные данные**

2 2