Problem B. Brackets and Dots

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

Андрей любит строки, которые состоят из круглых скобок, но еще больше он любит заменять скобки на точки.

У него есть строка длины N, к которой он применяет M операций. Каждая операция задается двумя натуральными числами l_i и r_i, в результате операции в подстроке $s_{l_i}s_{l_i+1}, \ldots, s_{r_i}$ самая длинная правильная скобочная подпоследовательность (далее ПСП) заменяется на точки (каждая скобочка заменяется на точку).

Андрей хочет себя проверить и просит вас помочь ему: подсчитать сколько скобок после каждой операции были заменены на точки.

Если в подстроке есть несколько ПСП максимальной длины, то выбирается наименьшая. Правило сравнения следующие: пусть $a_1, a_2, \ldots, a_l$ — позиции открывающихся скобок первой ПСП, а $c_1, c_2, \ldots, c_l$ — позиции открывающихся скобок второй ПСП. Считается, что первая ПСП меньше второй, если существует k $(1 \le k \le l)$, такое что $a_i = c_i$ $(1 \le i < k)$ и $a_k > c_k$.

Если позиции открывающихся скобок обоих ПСП совпали, то сравниваются позиции закрывающихся скобок по следующему правилу: пусть $b_1, b_2, \ldots, b_l$ — позиции закрывающихся скобок первой ПСП, а $d_1, d_2, \ldots, d_l$ — позиции закрывающихся скобок второй ПСП. Считается, что первая ПСП меньше второй, если существует k $(1 \le k \le l)$, такое что $b_i = d_i$ $(1 \le i < k)$ и $b_k < d_k$.

Input

На первой строке задается исходная строка S длины N $(1 \le N \le 5 \times 10^5)$, которая состоит из открывающихся и закрывающихся круглых скобок.

На второй строке задается натуральное число M $(1 \le M \le 5 \times 10^5)$ — количество операций.

В следующих M строках задаются операции, каждая строка состоит из двух натуральных чисел l_i и r_i $(1 \le l_i \le r_i \le N)$ — левая и правая граница подстроки соответственно.

Output

Для каждой операции на отдельной строке выведите единственное число — число скобок, которые были заменены на точки.

Example

стандартный ввод	стандартный вывод
((((())()))) 3 4 9 2 8 1 10	4 2 4

Problem C. Crossword

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	2 секунды
Memory limit:	256 мебибайт

Составление кроссвордов — непростая задача. Помимо того, чтобы соблюсти баланс по сложности слов, необходимо уместить слова в ограниченную площадь.

Но и это не всё! Пустого места должно быть не слишком много, но и не слишком мало, чтобы его можно было заполнить рекламой.

Вам даны четыре слова. Сколькими способами можно составить из них кроссворд, чтобы внутри образовалась пустая область в виде прямоугольника, ни одна из сторон которого не равна 0? Каждое слово должно пересекаться с ровно двумя другими словами.

Input

Ввод содержит 4 слова, находящиеся в отдельных строках.

Каждое слово имеет длину от 3 до 100 и состоит только из малых букв латинского алфавита.

Все слова различны.

Output

Выведите искомое количество вариантов.

Examples

стандартный ввод	стандартный вывод
aaa axa aya aza	24
aaaa abba baab bbbb	40

Problem D. Digit

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

Проснувшись ранним утром, Сашка понял, что давно не проведывал своего старого друга - Витальку. Решив не откладывать это дело на следующий день, он начал собираться в гости. Сашка уже стоял на пороге своей квартиры, как понял, что у него нет подарка! Зная слабость Витальки к цифрам, он решил подарить ему самую часто встречающуюся цифру в записи чисел от 1 до N, причем если таких несколько, было решено взять максимальную. Сможете ли Вы определить, какую цифру подарил Сашка своему другу?

Input

В единственно строке задано целое число N ($1 \le N \le 10^{100000}$).

Output

Выведите цифру, которую подарил Сашка.

Examples

стандартный ввод	стандартный вывод
100	1
99	9

Problem E. Enormous Table

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

В бесконечной таблице по номеру строки и столбца определить элемент.

```
1  2  6  7  15 ...
3  5  8  14 ...
4  9  13 ...
10 12 ...
11 ...
```

Input

Два натуральных числа a, b — номер строки и столбца соответственно. Числа не превосходят 10^9.

Output

Выведите одно натуральное число — ответ на задачу.

Example

стандартный ввод	стандартный вывод
2 3	8

Problem G. Game of Tic-Tac-Toe

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

Это интерактивная задача.

В недалёком будущем OpenAI бот уже выигрывает у человека почти во всех играх. Чтобы не дать боту захватить мир вам нужно написать программу, которая даст отпор при игре в классические крестики-нолики.

Напомним правила этой игры: игроки ходят по очереди, выставляя на клетчатом поле 3 на 3 свой символ (`X` или `O` — заглавные латинские буквы). Ходить можно только в свободную клетку. Если после хода образовалось 3 одинаковых символа в ряд (по вертикали, горизонтали или диагонали), игрок, которому принадлежит символ, выигрывает. Если такого не произошло, но при этом нет свободных клеток, то в игре объявляется ничья.

OpenAI бот всегда играет за нолики, Вы — за крестики. Ваша цель — не проиграть боту, glhf!

Interaction Protocol

В самом начале в первой строке ввода задаётся, кто ходит первым.

Далее происходит общение. Если ход за ботом, то он сообщает вам координаты клетки, в которую сделал ход, в виде пары целых чисел r и c, означающих номер строки и столбца. Строки нумеруются от 1 до 3 сверху вниз, а столбцы от 1 до 3 слева направо. Каждый ход выводится в отдельной строке, числа разделены одним пробелом. Гарантируется, что ходы бота корректны.

Если ход за вами, то Вы должны вывести в аналогичном формате координаты клетки хода.

Как только игра заканчивается, вам немедленно сообщается результат игры в виде строки «`WIN`» (вы выиграли), «`LOSE`» (вы проиграли) или «`DRAW`» (ничья).

Если вы проиграли или один из ваших ходов является некорректным, вам будет засчитан неправильный ответ на тест.

Examples

стандартный ввод	стандартный вывод
X 1 3 2 3 WIN	1 1 2 1 3 1
O 1 1 3 1 2 3 1 2 2 2 DRAW	3 3 1 3 2 1 3 2

Note

Для корректной работы программы после каждой операции вывода данных вам необходимо выводить перевод строки, а также очищать буфер вывода, то есть делать следующие операции:

- В языке Pascal: **flush(output)**;
- В C/C++: **fflush(stdout)** или **cout.flush()**;

- В Java: **System.out.flush**();
- В Python: **sys.stdout.flush**() из библиотеки **sys**;
- В C#: **Console.Out.Flush**();

Problem I. It is panic?

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

Ваш одногруппник Вася устроился в службу экстренного реагирования МЧС. Когда в городе что-то происходит, люди с места происшествия отправляют в эту службу SMS-сообщения. Если сообщение имеет вид «`AAA...!!!!`», то есть его можно разбить на две непустые половинки так, что левая состоит только из заглавных букв `A`, а правая — только из восклицательных знаков, то оно считается *паническим*, и на место этого происшествия группа реагирования отправляется в первую очередь. Если сообщение не паническое, то группа реагирования отправляется на место этого происшествия только тогда, когда нет других происшествий с паническими сообщениями.

В октябре как всегда внезапно выпал снег, в связи с этим количество происшествий, а значит, и SMS-сообщений, которые нужно обработать Васе, резко увеличилось. Помогите Васе и напишите программу, которая определяет, является SMS-сообщение паническим или нет.

Input

Ввод содержит непустую строку из заглавных и строчных символов латинского алфавита, а также знаков `!` и `?`. Длина строки не превышает 100.

Output

Выведите «`Panic!`» (без кавычек), если сообщение паническое, или «`No panic`» (без кавычек), если оно не паническое.

Examples

стандартный ввод	стандартный вывод
AAA!!!	Panic!
AAA	No panic
aaa!!!	No panic

Problem J. JokeCoin

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	2 секунды
Memory limit:	256 мебибайт

Шахтёр Василий заинтересовался криптовалютой SillyCoin. Майнеры этой валюты получают расписание блоков на следующие сутки. Для каждого блока известен промежуток времени, в течение которого его потребуется майнить, и вознаграждение в монетках SillyCoin, которое гарантированно выдаётся, если этот блок непрерывно майнился в указанное время. Майнер решает сам, стоит ли браться за майнинг того или иного блока. Майнинг следующего блока можно начинать в ту же секунду, как закончена обработка предыдущего блока. У Василия всего одного видеокарта, поэтому он не может майнить несколько блоков одновременно. Кроме того, Василию регулярно приходят счета за электричество, которое придётся оплачивать за фактически потраченное на майнинг время по фиксированному тарифу. Василий хочет узнать, выгодно ли ему заниматься майнингом, поэтому он просит вас написать программу, вычисляющую по заданному расписанию блоков максимально возможную выгоду.

Input

В первой строке находятся два целых числа N и C ($1 \le N < 86\,400, 0 \le C \le 1000$), разделённых пробелом, — количество блоков в расписании и стоимость электричества, затрачиваемого за каждую секунду майнинга, в монетках. В следующих N строках содержится расписание блоков на следующие сутки.

В каждой строке через пробел указаны время начала и окончания блока (в формате `HH:MM:SS` от `00:00:00` до `23:59:59` с разницей минимум в 1 секунду) и вознаграждение P ($0 \le P \le 10^5$) в монетках.

Output

Вывести целое число — максимальное вознаграждение, которое Василий может получить за майнинг. Если майнинг принесёт лишь убытки, следует вывести 0.

Examples

стандартный ввод	стандартный вывод
4 0 03:00:00 10:10:00 20 01:00:00 02:30:00 50 16:10:00 19:00:00 100 02:30:00 22:00:00 200	250
3 1 16:59:00 17:00:00 100 01:01:01 01:01:11 20 12:00:00 13:00:00 3601	51
4 10 00:00:05 00:01:55 1100 00:00:10 00:00:21 100 00:01:50 00:02:00 80 23:59:00 23:59:05 40	0

Problem K. King and ICPC

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	2 секунды
Memory limit:	256 мебибайт

Тёмные времена нависли над ACM-ом. Тысячи команд ACM-щиков вышли на защиту от нападения CTF-щиков. Всего имеется N команд, пронумерованных от 1 до N. Каждая команда, по традиции, состоит из трёх человек. Каждому человеку присвоили параметр ACM-овости, выраженный целым числом.

Король ACM-щиков хочет создать армию, выбрав из каждой команды ровно по одному человеку так, чтобы суммарная ACM-овость была как можно больше, но при этом делилась нацело на число D.

Так как ACM-щики по своей натуре любят усложнять задачи, они решили рассмотреть несколько отрезков $[l_i, r_i]$, составляя армию лишь из команд с номерами из этого отрезка.

Король поручил вам обработку этих запросов. На горизонте уже видны флаги, у вас осталось меньше пяти часов!

Input

Первая строка содержит два целых числа N ($1 \le N \le 50\,000$) и D ($1 \le D \le 50$).

В следующих N строках содержится описание каждой команды в виде трёх целых чисел. ACM-овость каждого человека является целым числом от 0 до 10^9.

Следующая строка содержит целое число M — количество отрезков ($1 \le M \le 300\,000$).

В следующих M строках идёт описание каждого запроса в виде пары чисел l_i и r_i ($1 \le l_i \le r_i \le N$).

Output

На каждый отрезок выведите в отдельной строке максимальную сумму, которую можно на нём собрать. Если же сумму, делящуюся на D собрать невозможно, выведите «-1» (без кавычек).

Examples

стандартный ввод	стандартный вывод
2 2 0 1 3 1 2 3 3 1 1 2 2 1 2	0 2 6
3 3 0 3 6 1 4 7 1 2 3 4 1 1 1 2 1 3 2 2	6 -1 15 -1

Problem M. Math Task

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

У Буратино было N яблок. Некто дал Буратино A яблок. Буратино съел B яблок. Сколько яблок осталось у Буратино?

Input

Два натуральных числа A и B ($1 \le A, B \le 10^{15}$)

Output

В общем виде выведите сколько яблок осталось у Буратино, при этом опускайте незначащую часть, если она есть.

Examples

стандартный ввод	стандартный вывод
2 1	N+1
2 3	N-1

Problem N. Naive HTML

Input file:	*стандартный ввод*
Output file:	*стандартный вывод*
Time limit:	1 секунда
Memory limit:	256 мебибайт

Разметка HTML содержит ноль или более вложенных элементов и текст (в рамках условия дана упрощенная версия реального HTML — Naive HTML, и он всегда корректен):

```
1.    <html>
2.        <input type="button" id="x" />
3.        <div id="x"class='cls'p="<i/>">
4.            hello <b>world</b> id='x'
5.        </div>
6.    </html>
```

HTML-код может содержать символы английского алфавита, символы "<>=/", а также кавычки ' и ". Также так называемые пробельные символы: пробелы, переводы строк и табуляцию. Все входные символы в задаче будут в нижнем регистре.

В данном примере веб-страница содержит элементы, обозначаемые тегами <html>, <input>, , <b>, а также текст "hello", "world" и "id='x'" внутри элементов. Теги состоят из объявления тега и опционально, закрытия тега. Рассмотрим тег tag:

- Объявление тега состоит из начала объявления – "<tag" и конца объявления — "/>" либо ">". Между началом и концом объявления может находиться ноль или более пробельных символов, а также атрибуты тега (см. ниже).
- Если тег содержит внутри себя другие теги или текст длиной ноль или более символов (как <html>, <b> или из примера), то конец объявления тега — это ">". Такой тег называют открытым. Если не содержит (как тег "<input>" из примера), то конец объявления тега - это "/>", и такой тег называют замкнутым.
- Открытый тег всегда закрывается как "</tag>" — как на строчках 4, 5, 6. Замкнутые теги не закрываются.
- между началом и концом объявления тега могут находиться атрибуты. Каждый атрибут имеет значение. Например, "<input>" имеет два атрибута — "type" со значением "button" и "id" со значением "x". Атрибуты записываются как "attr="val"" или "attr='val'" — закрывает значение та же кавычка, что и открывает. Название атрибута и его значение разделяется только символом '=' без пробелов. Название атрибута — один или более символов английского алфавита в нижнем регистре. На значения атрибутов накладывается единственное ограничение - они не могут содержать открывающую/закрывающую кавычку значения атрибута. Между атрибутами, а также атрибутами и началом объявления тега, атрибутами и концом объявления тега может быть ноль или более пробельных символов.

Вам нужно написать программу, которая по заданному HTML-коду и трем типам селекторов выдаст количество элементов на странице, соответствующих селекторам:

- "html" — возвращает все элементы <html> со страницы.
- "#myid" — возвращает все теги, где имеется атрибут id со значением myid.
- ".cls" — возвращает все теги, где имеется атрибут "class" со значением "cls".

Input

Первая строка содержит число n - количество селекторов (как минимум один). Далее следуют n строк без пробелов в начале и конце, которые обозначают селекторы $s_1, \ldots, s_n$, по которым Вам нужно сделать запросы. Селекторы содержат один или более символов английского алфавита и/или символы "`.#`'. Далее до конца входа находится непустой HTML-код. Общая длина входных данных не превышает 5 000 символов. Все символы во входе находятся в нижнем регистре.

Output

Для каждого запроса $s_1, \ldots, s_n$ напечатайте ответ на отдельной строке, содержащий селектор и количество элементов HTML-страницы, ему соответствующих. См. пример выходных данных.

Example

стандартный ввод	стандартный вывод
5 html #x .cls #fakeid i <html> <input type="button" id="x" /> <div id="x"class='cls'p="<i/>"> hello world id='x' </html>	Selector "html": found 1 elements Selector "#x": found 2 elements Selector ".cls": found 1 elements Selector "#fakeid": found 0 elements Selector "i": found 0 elements

Note

Обратите внимание, что в качестве примера входных данных используется HTML-страница из условия.