Задача A. Логи Apache

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

8 октября в Центр была передана информация от Штирлица — лог сервера Apache, отражающий доступ к ключевому интернет-ресурсу, расположенному в государстве Borderland. В результате успешной спецоперации в Borderland сложилась тупиковая ситуация вокруг референдума о том, какая из букв нового алфавита должна быть первой. После предыдущих итераций выбор сузился до `Y` и `Z`. Очередная итерация процесса выхода из кризиса намечалась уже через пять дней. Для того чтобы восстановить последовательность запросов, сотрудники Центра, не совсем пришедшие в себя после Дня Чекиста, решили перевести события лога к временной зоне секретной базы, на которой разрабатывается новая диверсия.

Формат входных данных

В первой строке записана временная зона секретной базы, к которой нужно преобразовать лог. Начиная со второй строки, задан сам лог. Строки лога Apache всегда имеют вид

$$\langle ip \rangle \ - \ \langle username \rangle \ \langle date \rangle \ \langle extra\ fields \rangle$$

Поле *ip* представляет собой ip-адрес узла, сделавшего запрос, в стандартном формате. Второе поле всегда содержит один символ «-». Имя пользователя состоит из произвольного ненулевого количества латинских букв, символов «-» и кавычек. Формат даты следующий:

$$[\langle день \rangle/\langle месяц \rangle/\langle год \rangle:\langle часы \rangle:\langle минуты \rangle:\langle секунды \rangle \ \langle временная\ зона \rangle]$$

Временная зона всегда записывается ровно пятью символами:

$$\langle знак \rangle \langle часы \rangle \langle минуты \rangle$$

Знак принимает значение «+», если часы в данной временной зоне опережают часы в зоне GMT (Greenwich Mean Time) или показывают то же время, в противном случае ставится знак «-».

День, часы, минуты и секунды всегда состоят ровно из двух цифр (как и в поле даты, так и во всех записях временных зон), год — из четырех. Месяц записан сокращением на английском языке и может принимать одно из следующих значений: `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov`, `Dec`.

Допустимые во входных данных номера года лежат в диапазоне от 1980 до 2099. Високосными считаются года, номер которых делится на 4.

Дополнительные поля состоят из произвольного количества символов с кодами от 32 до 126.

В записях временной зоны поле часов всегда содержит значение, не превосходящее 12. Размер входных данных не превосходит одного мегабайта. Все даты корректны и проверка их не требуется.

Формат выходных данных

Результатом работы программы является лог Apache, в котором все времена и даты преобразованы к заданной временной зоне. Все остальные поля в каждой строке лога должны оставаться без изменений. Имейте в виду, что проверка производится автоматически, поэтому все даты и времена должны содержать по две цифры в полях дней, часов, минут и секунд, три символа в поле месяца (case-sensitive), и четыре цифры в поле года.

Пример

стандартный ввод

```
-0200
195.19.224.104 - abc [02/Dec/2004:18:25:19 +0100] "GET / HTTP/1.1" 304 -
195.19.224.104 - - [02/Dec/2004:18:25:19 +0100] "GET /main.css HTTP/1.1" 304 -
195.19.224.79 - - [02/Dec/2004:20:58:20 +0300] "GET /tts HTTP/1.1" 302 293
195.19.224.79 - - [02/Dec/2004:20:58:20 +0300] "GET /tts/ HTTP/1.1" 302 303
207.46.98.42 - - [03/Dec/2004:04:39:32 +0300] "GET /robots.txt HTTP/1.0"
```

стандартный вывод

```
195.19.224.104 - abc [02/Dec/2004:15:25:19 -0200] "GET / HTTP/1.1" 304 -
195.19.224.104 - - [02/Dec/2004:15:25:19 -0200] "GET /main.css HTTP/1.1" 304 -
195.19.224.79 - - [02/Dec/2004:15:58:20 -0200] "GET /tts HTTP/1.1" 302 293
195.19.224.79 - - [02/Dec/2004:15:58:20 -0200] "GET /tts/ HTTP/1.1" 302 303
207.46.98.42 - - [02/Dec/2004:23:39:32 -0200] "GET /robots.txt HTTP/1.0"
```

Задача B. Деление уголком

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Вы отправились на машине времени в далёкое прошлое Франции с важной исторической миссией. Для укрепления своего положения в обществе вам необходимо получить учёную степень.

Явившись в ближайший университет, вы с удивлением обнаружили, что для этого необходимо всего лишь продемонстрировать умение делить числа уголком. Для вас это просто, но для стопроцентной гарантии вы решили попросить компьютер проделать аналогичные вычисления.

Напишите программу, которая выводит процесс деления двух десятичных чисел уголком.

Формат входных данных

В двух строках входных данных заданы делимое и делитель, меньшие 10^{100}.

Формат выходных данных

Выведите процесс деления. В точности следуйте формату примера. Никакие числа в процессе вывода не могут иметь ведущие нули. Сравнение будет производиться посимвольно. Делимое должно быть отделено от вертикальной черты ровно одним пробелом.

Примеры

стандартный ввод	*стандартный вывод*

50082003928
491
	50082003928 |491
	491 +---------
	----- |102000008
	  982
	  982
	  --------- 	
	  3928
	  3928
	  ---- 	
	  0

239
717
	239 |717
	  +---
	  |0

239 17	239 |17 17 +-- --- |14 69 68 -- 1
667700 6677	667700 |6677 6677 +---- ------ |100 0
12 7	12 |7 7 +- -- |1 5

Задача C. Сообщение об ошибке

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Правильные скобочно-палочные последовательности (ПСПП) определяются рекурсивно следующим образом:

1. Пустая строка является ПСПП.
2. Если A и B — две ПСПП, то AB (строка, полученная приписыванием строки B справа к строке A) — тоже ПСПП.
3. Если A и B — две ПСПП, то строка `(`A`|`B`)` также является ПСПП.

Например, «`(|)`» является ПСПП, так как получается по пункту 3 определения для пустых строк A и B, которые являются ПСПП по пункту 1. Строка «`((|)|)(|)`» также является ПСПП: по пункту 2 её можно составить из строк $A =$ `((|)|)` и $B =$ `(|)`. В свою очередь, строка «`((|)|)`» является ПСПП по пункту 3 определения для $A =$ `(|)` и пустой строки B.

Задана строка, состоящая из символов «`(`», «`|`» и «`)`». Если эта строка является ПСПП, выведите сообщение «`correct, length =` x», где x — длина строки. В противном случае выведите сообщение об ошибке вида «`at position` p`: expected` c_1`[ or` c_2`[ or` c_3`]], found` e». Такое сообщение означает, что первые $(p-1)$ символов строки являются ПСПП или началом какой-либо ПСПП, а на следующей позиции должен оказаться один из вариантов c_1, c_2 или c_3, чтобы строка была ПСПП или началом какой-либо ПСПП. Вместо этого на p-й позиции строки оказался вариант e. Позиции в строке нумеруются с единицы.

Во втором сообщениии вместо каждого из обозначений вариантов — c_1, c_2, c_3 и e — стоит либо один из трёх возможных символов «`(`», «`|`» и «`)`», либо строка «`END`», обозначающая конец строки. Возможные значения упорядочены следующим образом: сначала «`(`», затем «`|`», далее «`)`» и, наконец, «`END`». Значения в списке c_i должны быть перечислены в соответствующем этому порядке.

Квадратные скобки означают, что часть сообщения, заключённая в них, может присутствовать, а может отсутствовать: например, если в какой-то позиции есть ровно два правильных варианта, сообщение выглядит как «`at position` p`: expected` c_1 `or` c_2`, found` e».

Формат входных данных

Единственная строка ввода имеет длину от 1 до 60 символов и состоит исключительно из символов «`(`», «`|`» и «`)`» (ASCII-коды 40, 124 и 41).

Обратите внимание: после всех этих символов следует перевод строки.

Формат выходных данных

Выведите сообщение о первой неправильной позиции в строке или о том, что строка является ПСПП. Сообщение должно иметь формат, указанный в условии. Пожалуйста, соблюдайте его как можно точнее.

Примеры

стандартный ввод	*стандартный вывод*
`(|(|))`	`correct, length = 6`
`(|))|`	`at position 4: expected ( or END, found )`
`((`	`at position 3: expected ( or |, found END`

Пояснения к примерам

В первом примере строка «`(|(|))`» целиком является ПСПП. Она получается по пункту 3 рекурсивного определения для пустой строки A и $B =$ `(|)`.

Во втором примере строка «`(|)`» или, например, «`(|)(|)`» была бы ПСПП. Однако нет ПСПП, начинающейся на «`(|))`». Из возможных вариантов открывающая скобка «`(`» должна быть выведена раньше конца строки «`END`».

В третьем примере строка могла бы продолжаться как «`((|)|)`» или «`(((|)|)|)`». Вместо этого на позиции 3 оказался конец строки, обозначаемый при выводе сообщения как «`END`». Из возможных вариантов открывающая скобка «`(`» должна быть выведена раньше вертикальной черты «`|`».

Задача D. Ординальные числа

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Ординальное число — это обобщение понятия натурального числа (здесь и далее в этой задаче 0 считается натуральным числом). Ординальные числа, соответствующие натуральным числам, можно определить рекурсивно:

> Ординальное число, соответствующее натуральному числу n — это множество, элементами которого являются ординальные числа, соответствующие натуральным числам 0, 1, 2, …, $n-1$.

В частности, пустое множество $\varnothing = \{\}$ — ординальное число, соответствующее натуральному числу 0, единице соответствует множество $\{\varnothing\} = \{\{\}\}$, двойке — множество $\{\varnothing, \{\varnothing\}\} = \{\{\}, \{\{\}\}\}$ и так далее. Удобно записывать ординальное число просто как соответствующее ему натуральное число, например, $4 = \{0, 1, 2, 3\}$. Вообще, $n = \{0, 1, \ldots, (n-1)\}$.

Такое определение удобно тем, что его можно расширить на бесконечные множества. К примеру, наименьшее бесконечное ординальное число ω определяется как множество, содержащее все конечные ординальные числа. Следующее за ним число (будем записывать его как $\omega + 1$) равно $\{\omega, 0, 1, \ldots\}$ или, другими словами, $\omega + 1 = \omega \cup \{\omega\}$, и так далее. В общем случае по любому ординальному числу α можно определить следующее ординальное число $\alpha + 1 = \alpha \cup \{\alpha\}$. Однако, предыдущее ординальное число определено не для всех чисел, например, для ω не существует такого γ, что $\omega = \gamma \cup \{\gamma\}$.

Игорь выписал на доске некоторое конечное ординальное число. Запись была корректной и состояла из открывающих и закрывающих фигурных скобок, а также запятых. Формат записи рекурсивно определяется так: запись каждого множества начинается с открывающей фигурной скобки, далее по одному разу следуют записи всех элементов этого множества в произвольном порядке, разделённые запятыми, а в конце стоит закрывающая фигурная скобка. К примеру, ординальное число 0 записывается как «`{}`», 1 — как «`{{}}`», 2 — как «`{{},{{}}}`» или «`{{{}},{}}`» и так далее.

Улучив момент, Владислав подкрался к доске и стёр один символ. Запись больше не является корректной! Это несколько расстраивает Игоря. Установив виновника, Владимир Михайлович попросил Владислава срочно исправить положение. Помогите Владиславу восстановить исходную запись.

Формат входных данных

В первой строке ввода задана запись ординального числа, в которой отсутствует ровно один символ. Длина этой записи — от 1 до $3 \cdot 10^6$ символов.

Формат выходных данных

Выведите одну строку — правильную запись ординального числа, полученную добавлением одного символа в любое место заданной записи. Если правильных ответов несколько, выведите любой из них.

Примеры

стандартный ввод	*стандартный вывод*
`{{{}}{}}`	`{{{}},{}}`
`{{{}},}}`	`{{{}},{}}`
`{{{},{}}`	`{{{}},{}}`

Пояснения к примерам

Во всех трёх тестах после добавления одного символа получается ординальное число 2.

В первом тесте нужно добавить запятую, во втором — открывающую фигурную скобку, в третьем — закрывающую фигурную скобку.

Задача E. Таймер

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Рассмотрим следующую конструкцию таймера. Таймер состоит из двух цилиндров: внутреннего и внешнего, который может вращаться вокруг внутреннего. На поверхность внутреннего цилиндра нанесены отметки и числа. Отметки соответствуют минутам. Внешний же цилиндр покрывает весь внутренний, оставляя видимым только маленькое окошко. Время, которое показывает таймер, можно узнать, посмотрев в центр этого окошка.

Всего на внутреннем цилиндре 60 отметок. Каждая пятая из них отмечена соответствующим числом: 0, 5, 10, 15, …, 55. Расстояния между любыми двумя соседними отметками, включая расстояние между последней и первой отметками, равны. Ширина окошка ровно в пять раз больше расстояния между соседними отметками.

Исходно в центре окошка во внешнем цилиндре находится отметка 0 на внутреннем цилиндре. Чтобы выставить время на таймере, внешний цилиндр поворачивают вокруг внутреннего по часовой стрелке. После этого он медленно поворачивается против часовой стрелки, пока опять не достигнет отметки 0, после чего останавливается. На таймере можно выставить любое время, строго меньшее, чем 60 минут.

Анна учит своего домашнего робота Берту узнавать время, которое показывает таймер. Берта может фотографировать окошко таймера. Помогите Анне написать программу для Берты, чтобы определять время по фотографии.

Фотография окошка — это растровое изображение 60 пикселов в ширину и 12 пикселов в высоту. Будем для простоты считать, что фотография — это идеальное изображение видимой части внутреннего цилиндра, уложенной на плоскость. В таком случае расстояние между соседними отметками будет ровно 12 пикселов. Также для простоты предположим, что фотография сделана в момент, когда время, которое показывает таймер, кратно 5 секундам. Поскольку окошко не совсем прямоугольной формы, небольшие уголки на фотографии всегда закрыты внешним цилиндром; точную форму уголков можно увидеть в примере. Все остальные пикселы фотографии либо полностью белые, либо полностью чёрные.

При указанных предположениях каждая отметка выглядит как чёрный квадрат 2×2 в первых двух строках фотографии; настоящий центр отметки находится между двумя столбцами этого квадрата. Каждое число печатается чёрным шрифтом размера 8×8 в строках с 4-й по 11-ю. Каждое число (из одной или двух цифр) горизонтально выровнено так, что центр соответствующего блока размера 8×8 или 16×8 расположен по центру отметки, при которой написано это число. Все пикселы, не покрытые уголками, отметками или цифрами, отображаются как белые.

Ниже показаны изображения отдельных цифр от 0 до 9 в используемом шрифте 8×8. Сам шрифт идентичен шрифту, использовавшемуся в видеокартах *Rendition Vérité 1000* в 1990-е годы. Изображения цифр в шрифте можно также скачать по следующему адресу:
`http://acm.math.spbu.ru/trains/191022_m19/timer/`.

```
.xxxxx.. ...xx... .xxxxx.. .xxxxx.. ...xxx.. xxxxxxx. .xxxxx.. xxxxxxx. .xxxxx.. .xxxxx..
xx..xxx. ..xxx... xx...xx. xx...xx. ..xxxx.. xx...... xx...xx. xx...xx. xx...xx. xx...xx.
xx.xxxx. .xxxx... .....xx. .....xx. .xx.xx.. xxxxxx.. xx...... x....xx. xx...xx. xx...xx.
xxxx.xx. ...xx... ...xxx.. ..xxxx.. xx..xx.. .....xx. xxxxxx.. ....xx.. .xxxxx.. .xxxxxx.
xxx..xx. ...xx... .xxx.... .....xx. xxxxxxx. .....xx. xx...xx. ...xx... xx...xx. .....xx.
xxx..xx. ...xx... xx...... xx...xx. ....xx.. xx...xx. xx...xx. ..xx.... xx...xx. xx...xx.
.xxxxx.. .xxxxxx. xxxxxxx. .xxxxx.. ...xxxx. .xxxxx.. .xxxxx.. ..xx.... .xxxxx.. .xxxxx..
........ ........ ........ ........ ........ ........ ........ ........ ........ ........
```

Формат входных данных

Входные данные состоят ровно из 12 строк, каждая из которых содержит ровно 60 символов: фотография окошка таймера. Поскольку окошко не совсем прямоугольной формы, небольшие уголки на фотографии всегда закрыты символами «-» (знак минус); точную форму уголков можно увидеть в примере. Все остальные символы соответствуют изображению внутреннего цилиндра и равны либо «.» (точка) для белых пикселов, либо «X» (большая буква икс) для чёрных пикселов.

Гарантируется, что таймер корректно показывает некоторое время, строго меньшее 60 минут и кратное 5 секундам.

Формат выходных данных

Выведите время, которое показывает таймер, на отдельной строке в формате «MM:SS». Здесь «MM» — две цифры, соответствующие минутам, а «SS» — две цифры, соответствующие секундам.

Пример

стандартный ввод
-------X.........XX.........XX.........XX.........------- ----..XX.........XX.........XX.........XX.........XX---- -...- X...XXXXXXX.. XX..XX....... XX..XXXXXX... XX...XX.. XX...XX.. XX..XX...XX.. -...XXXXX..- ----...---- -------...-------
стандартный вывод
07:05

Пояснение к примеру

В этом примере центр окошка находится между отметками 10 и 5. Отметка, обозначенная числом 5, хорошо видна справа. Часть нуля из числа 10 можно увидеть слева. Более тщательное изучение фотографии позволяет установить, что время, которое показывает таймер, в точности равно 7 минутам и 5 секундам.

Задача F. Различные подстроки 3

Имя входного файла:	*стандартный ввод*
Имя выходного файла:	*стандартный вывод*
Ограничение по времени:	2 секунды
Ограничение по памяти:	256 мебибайт

Подстрокой строки $\mathcal{S} = s_1 s_2 \dots s_n$ называется непрерывная подпоследовательность символов этой строки $s_i s_{i+1} s_{i+2} \dots s_{j-1} s_j$.

Дана строка. Сколько различных подстрок, не считая пустой, она содержит? Выведите все эти подстроки по одному разу в лексикографическом порядке. Рядом с каждой подстрокой выведите, сколько раз она встречается в строке $\mathcal{S}$.

Формат входных данных

В первой строке ввода задана строка длины от 1 до 100 символов, включительно. Строка состоит из маленьких букв английского алфавита.

Формат выходных данных

В первой строке выведите одно число r — количество различных подстрок данной строки, не считая пустой. В следующих r строках выведите сами эти подстроки в лексикографическом порядке. После каждой подстроки через пробел должно следовать целое число — сколько раз эта подстрока встречается в строке $\mathcal{S}$.

Примеры

стандартный ввод	*стандартный вывод*
aab	5 a 2 aa 1 aab 1 ab 1 b 1
da	3 a 1 d 1 da 1