Задача A. Нормализация лямбда-выражения

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	15 секунд
Ограничение по памяти:	1024 мегабайта

Дано лямбда-выражение, требуется провести m ($m \in \mathbb{N}_0$) бета-редукций этого выражения используя нормальный порядок редукции и мемоизацию, при этом выводить на печать требуется каждое k-е выражение ($k \in \mathbb{N}_0, k < m$). Формулы нумеруются с 0, если нормальная форма была достигнута на формуле с некратным k номером — на формуле δ_s, где $k \cdot (n-1) < s < k \cdot n$, — то выдача должна завершиться формулой δ_s. Например, редуцирование выражения $(\lambda x.x\ x\ x\ x)\ ((\lambda x.x)\ (\lambda x.x))$ в данных условиях пройдёт через следующие стадии (редуцируемые бета-редексы подчёркнуты):

обозначение (номер)	формула
δ_0	$\underline{(\lambda x.x\ x\ x\ x)\ ((\lambda x.x)\ (\lambda x.x))}$
δ_1	$\underline{((\lambda x.x)\ (\lambda x.x))}\ \underline{((\lambda x.x)\ (\lambda x.x))}\ \underline{((\lambda x.x)\ (\lambda x.x))}\ \underline{((\lambda x.x)\ (\lambda x.x))}$
δ_2	$\underline{(\lambda x.x)\ (\lambda x.x)}\ (\lambda x.x)\ (\lambda x.x)$
δ_3	$\underline{(\lambda x.x)\ (\lambda x.x)}\ (\lambda x.x)$
δ_4	$\underline{(\lambda x.x)\ (\lambda x.x)}$
δ_5	$(\lambda x.x)$

Если при этом $k = 2$, то на печать должны быть выведены формулы δ_0, δ_2, δ_4, δ_5.

Гарантируется, что суммарная длина всех выражений, которые будут получены в результате s бета-редукций, не превышает 100 миллионов лексем.

Для точного определения условий задачи, давайте напомним два важных определения — нормальный порядок редукций и мемоизацию.

1. Рассмотрим лямбда-выражение, расставим все необязательные скобки в нём. Назовём нормальным порядком редукции такой порядок, при котором всегда редуцируется самый левый редекс: то есть редекс, первый символ которого находится левее всего в выражении.

2. Чтобы определить мемоизацию, определим некоторое расширенное лямбда-исчисление. Помимо обычных выражений будем рассматривать отложенные подстановки: это переменные с указанием заменяемого выражения в угловых скобках — $x_{\langle A \rangle}$.

   При этом подстановка $A[x := B]$ раскрывается так:

$$A[x := B] = \begin{cases} t_{\langle B \rangle}, & A = x \\ y, & A = y, y \neq x \\ \lambda x.P, & A = \lambda x.P \\ \lambda y.(P[x := B]), & A = \lambda y.P, y \neq x \\ (P[x := B])\ (Q[x := B]), & A = P\ Q \end{cases}$$

   Здесь t — некоторая новая отложенная переменная, ранее в выражении не встречавшаяся.

   Естественным образом мы можем определить плоское лямбда-выражение для данного выражения, рассматривая каждую переменную вида $x_{\langle P \rangle}$ как P.

   Тогда шаг редукции с мемоизацией устроен так:

   - Выберем редекс $(\lambda x.A)\ B$ — например, найдём самый левый редекс в плоском лямбда-выражении, соответствующем данному.
   - Если $(\lambda x.A)$ содержит вхождение отложенной подстановки $y_{\langle P \rangle}$, в которую входит заменяемая переменная x, перед редукцией заменим данное вхождение $y_{\langle P \rangle}$ на P. Обратите внимание, случай $\lambda x.A = y_{\langle P \rangle}$ также надо учитывать.
   - Все остальные отложенные подстановки в редексе оставим без изменений — рассматриваем, как переменные. Производим редукцию.

- Если редекс целиком находится внутри какой-то отложенной подстановки — редукцию производим во всех отложенных подстановках по той же переменной.

Формат входных данных

В первой строке приведены числа m и k через пробел. Во второй строке дано лямбда-выражение δ_0 в грамматике из предыдущего задания.

Формат выходных данных

Выведите формулы δ_0, δ_k, $\delta_{k\cdot 2}$, ..., $\delta_{k\cdot(n-1)}$, δ_s, по формуле на новой строке.

Примеры

стандартный ввод	стандартный вывод
10 1 (\x.x) z	((\x.x) z) z
100 1 (\x.y) z	((\x.y) z) y
100 1 (\a.\a.b) c	((\a.(\a.b)) c) (\v0.b)
100 1 (\a.\x.a) (x y)	((\a.(\x.a)) (x y)) (\v0.(x y))

Задача B. Проверка вывода типа в системе Хиндли-Миллнера

Имя входного файла:	стандартный ввод
Имя выходного файла:	стандартный вывод
Ограничение по времени:	1 секунда
Ограничение по памяти:	256 мегабайт

На вход подается доказательство в следующей грамматике:

```
<Доказательство>     ::= <Строка>*

<Строка>             ::= <Отступ> <Контекст> <Выражение с типом> <Правило>

<Отступ>             ::= ('*' ' ' ' ' ' ' ')*

<Контекст>           ::= eps
                       | <Переменная> : <Тип> [, <Контекст>]

<Выражение с типом>  ::= <Выражение> : <Тип>

<Тип>                ::= (<Тип>)
                       | <Монотип>
                       | forall <Переменная> . <Тип>

<Монотип>            ::= <Переменная>
                       | (<Монотип>)
                       | (<Монотип>) -> <Монотип>
                       | <Переменная> -> <Монотип>

<Выражение>          ::= [<Применение>] \ <Переменная> . <Выражение>
                       | let <Переменная> = <Выражение> in <Выражение>
                       | <Применение>

<Применение>         ::= <Атом>
                       | <Применение> <Атом>

<Атом>               ::= (<Выражение>)
                       | <Переменная>

<Переменная>         ::= [a-z] [a-z0-9']*

<Правило>            ::= "[rule #" <Номер правила> "]"

<Номер правила>      ::= [1-6]
```

В доказательстве могут быть использованы следующие правила:

Правило	Зависимости	Вывод	Дополнительные условия
1		$\Gamma \vdash x : \sigma$	$x : \sigma \in \Gamma$, x — переменная
2	$\Gamma \vdash e_0 : \tau \to \tau' \quad \Gamma \vdash e_1 : \tau$	$\Gamma \vdash e_0 e_1 : \tau'$	τ, τ' — монотипы
3	$\Gamma, x : \tau \vdash e : \tau'$	$\Gamma \vdash \lambda x.e : \tau \to \tau'$	τ, τ' — монотипы, x — переменная
4	$\Gamma \vdash e_0 : \sigma \quad \Gamma, x : \sigma \vdash e_1 : \tau$	$\Gamma \vdash$ let $x = e_0$ in $e_1 : \tau$	τ — монотип, x — переменная
5	$\Gamma \vdash e : \sigma'$	$\Gamma \vdash e : \sigma$	$\sigma' \sqsubseteq \sigma$
6	$\Gamma \vdash e : \sigma$	$\Gamma \vdash e : \forall \alpha.\sigma$	$\alpha \notin$ free (Γ), α — типовая переменная

Назовём детьми строки доказательства, строки с отступом на 1 больше, идущие после неё и до первой строки после неё с отступом меньше или равным её отступу. Доказательство корректно, если для каждой из строк в доказательстве верно, что выражение в этой строке выводится по правилу, указанному в этой строке, из выражений в её детях, при чём дети должны идти в доказательстве в том же порядке, что указаны в таблице зависимостей, и после неё идёт строка с отступом не более чем её отступ плюс 1, либо конец доказательства.

Определите, корректное ли это доказательство.

Формат входных данных

Во входных данных даны несколько строк, являющихся доказательством в указанной грамматике. Отступы всегда выглядят как * и три пробела, но между любыми другими токенами могут быть пробельные символы.

Гарантируется, что в каждом контексте все переменные, для которых заданы типы, различны.

Гарантируется, что ключевые слова `let`, `in`, `forall` не будут склеены с названиями переменных и другими ключевыми словами.

Названия переменных не могут начинаться на ключевые слова `let`, `in`, `forall`.

Суммарная длина входных данных не превосходит 10^5 байт.

Формат выходных данных

Выведите `Correct`, если доказательство корректно, и `Incorrect` иначе.

Примеры

стандартный ввод	стандартный вы
x : a |- x : (forall b. a) [rule #6] * x : a |- x : a [rule #1]	Correct
|- ((\x. x) (\y. y)) : (t2 -> t3) [rule #2] * |- (\x. x) : ((t2-> t2)-> (t2 -> t2)) [rule #3] * * x : (t2 -> t2) |- x : (t2 -> t2) [rule #1] * |- (\y. y) : (t2 -> t2) [rule #3] * * y : t2 |- y : t2 [rule #1]	Incorrect
x : t1 |- x : t1 [rule #1]	Correct