Задача A. Расстановка скобок в лямбда-выражении

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

На вход вашей программе дается лямбда-выражение в следующей грамматике:

$$
\begin{array}{rcl}
\langle\text{Выражение}\rangle & ::= & [\langle\text{Применение}\rangle]\ \text{`\textbackslash'}\ \langle\text{Переменная}\rangle\ \text{`.'}\ \langle\text{Выражение}\rangle \\
& | & \langle\text{Применение}\rangle \\
\langle\text{Применение}\rangle & ::= & \langle\text{Применение}\rangle\ \langle\text{Атом}\rangle \mid \langle\text{Атом}\rangle \\
\langle\text{Атом}\rangle & ::= & \text{`('}\ \langle\text{Выражение}\rangle\ \text{`)'} \mid \langle\text{Переменная}\rangle \\
\langle\text{Переменная}\rangle & ::= & (\text{`a'}\ldots\text{`z'})\ \{\text{`a'}\ldots\text{`z'} \mid \text{`0'}\ldots\text{`9'} \mid \text{`'}\}^*
\end{array}
$$

Аргументы-переменные в применении должны разделяться пробелом. В остальных случаях пробелы могут отсутствовать. Любые пробелы между нетерминальными символами (кроме пробела, разделяющего аргументы в применении) — а также начальные и конечные пробелы в строке — должны игнорироваться. Символы табуляции, возврата каретки и перевода строки должны трактоваться как пробелы.

Требуется расставить все недостающие скобки вокруг всех абстракций и применений, и напечатать получившийся результат.

Формат входных данных

Размер входного файла не превышает 1 МБ.

Формат выходных данных

В единственной строке выходного файла (заканчивающейся переводом строки) должно быть приведено лямбда-выражение с расставленными скобками без каких-либо пробельных символов. Исключение: одиночные пробелы, разделяющие аргументы в применении.

Примеры

стандартный ввод
`\a.\b.a b c (\d.e \f.g) h`
стандартный вывод
`(\a.(\b.((((a b) c) (\d.(e (\f.g)))) h)))`

стандартный ввод
`((a\bbb.c)d)e` `f g`
стандартный вывод
`(((((a (\bbb.c)) d) e) f) g)`

Задача B. Нормализация лямбда-выражения

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	15 секунд
Ограничение по памяти:	512 мегабайт

Дано лямбда-выражение, имеющее нормальную форму. Требуется его нормализовать.

Формат входных данных

В единственной строке дано лямбда-выражение в грамматике из предыдущего задания. Гарантируется, что оно имеет нормальную форму.

Формат выходных данных

В единственной строке выведите нормализованное лямбда-выражение в той же грамматике.

Примеры

стандартный ввод	стандартный вывод
(\x.x) z	z
(\x.y) z	y
(\a.\a.b) c	\a.b
(\a.\x.a) (x y)	\x'.x y

Задача C. Вывод типа в просто-типизированном лямбда-исчислении

Имя входного файла:	`стандартный ввод`
Имя выходного файла:	`стандартный вывод`
Ограничение по времени:	5 секунд
Ограничение по памяти:	512 мегабайт

На вход вашей программе дается лямбда-выражение в следующей грамматике:

$$
\begin{array}{rcl}
\langle\text{Выражение}\rangle & ::= & [\langle\text{Применение}\rangle]\ \text{`\textbackslash'}\ \langle\text{Переменная}\rangle\ \text{`.'}\ \langle\text{Выражение}\rangle \\
& | & \langle\text{Применение}\rangle \\
\langle\text{Применение}\rangle & ::= & \langle\text{Применение}\rangle\ \langle\text{Атом}\rangle \mid \langle\text{Атом}\rangle \\
\langle\text{Атом}\rangle & ::= & \text{`('}\ \langle\text{Выражение}\rangle\ \text{`)'} \mid \langle\text{Переменная}\rangle \\
\langle\text{Переменная}\rangle & ::= & (\text{`a'}\ldots\text{`z'})\ \{\text{`a'}\ldots\text{`z'} \mid \text{`0'}\ldots\text{`9'} \mid \text{`'}\}^*
\end{array}
$$

Аргументы-переменные в применении разделяются пробелом. В остальных случаях пробелы могут отсутствовать. Любые пробелы между нетерминальными символами (кроме пробела, разделяющего аргументы в применении) — а также начальные и конечные пробелы в строке — должны игнорироваться. Символы табуляции и возврата каретки должны трактоваться как пробелы.

Требуется найти наиболее общий тип этого лямбда-выражения и вывести доказательство того, что лямбда-выражение имеет этот тип, а также найти типы свободных переменных, содержащихся в лямбда-выражении, или же сказать, что лямбда-выражение не имеет типа.

В доказательстве вы можете пользоваться следующими правилами:

Правило	Зависимости	Вывод	Дополнительные условия
№1		$\Gamma, x : \sigma \vdash x : \sigma$	$x \notin \text{dom}(\Gamma)$
№2	$\Gamma \vdash M : \sigma \to \tau, N : \sigma$	$\Gamma \vdash MN : \tau$	
№3	$\Gamma, x : \sigma \vdash M : \tau$	$\Gamma \vdash \lambda x.M : \sigma \to \tau$	$x \notin \text{dom}(\Gamma)$

Формат входных данных

В единственной строке входного файла содержится лямбда-выражение в грамматике из условия. Длина выражения не превышает 255 символов.

Гарантируется, что имена всех вложенных абстракций различны, а также имена абстракций не совпадают с именами свободных переменных.

Формат выходных данных

Если заданное лямбда-выражение не имеет типа, в единственной строке выходного файла должна быть запись «`Expression has no type`».

Иначе в файле должно быть доказательство. В файле должны отсутствовать пустые строки. Строки доказательства должны идти в правильном порядке. Каждый отступ должен представляться с помощью «`*  `» — символа «*» (ASCII 42) и трех последовательных пробелов (ACSII 32). В конце каждой строки должно быть описание правила, которое было применено для вывода этой строки. В остальном следуйте формату из примеров.

Выведенный тип должен быть наиболее общим типом для заданного лямбда-выражения.

Примеры

стандартный ввод
`x`
стандартный вывод
`x : t1 |- x : t1 [rule #1]`

стандартный ввод

```
(\x. x) (\y. y)
```

стандартный вывод

```
|- ((\x. x) (\y. y)) : (t2 -> t2) [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]
```

стандартный ввод

```
\a. a' a z8'
```

стандартный вывод

```
a' : (t1 -> (t4 -> t5)), z8' : t4 |- (\a. ((a' a) z8')) : (t1 -> t5) [rule #3]
*   a' : (t1 -> (t4 -> t5)), z8' : t4, a : t1 |- ((a' a) z8') : t5 [rule #2]
*   *   a' : (t1 -> (t4 -> t5)), z8' : t4, a : t1 |- (a' a) : (t4 -> t5) [rule #2]
*   *   *   a' : (t1 -> (t4 -> t5)), z8' : t4, a : t1 |- a' : (t1 -> (t4 -> t5)) [rule #1]
*   *   *   a' : (t1 -> (t4 -> t5)), z8' : t4, a : t1 |- a : t1 [rule #1]
*   *   a' : (t1 -> (t4 -> t5)), z8' : t4, a : t1 |- z8' : t4 [rule #1]
```