Lambda calculus P86813_en

Olimpiada Informática Española — Final 2007 (2007)

Your program has to read lambda expression and to do 3 basic operations of the lambda calculation: calculate the "spent" variables, calculate the free variables, and do substitutions.

Lambda expressions. The language of the lambda expressions (expressions from now on) is described with the following three cases and no one else:

- Strings of characteres from `'a'` to `'z'` are expressions (variables).
- Given an expression B and a variable v, the string formed by the character `'\'`, followed by v, followed by the character `'.'` followed B ($\backslash v\,.\,B$) is an expression (abstraction).
- Given two expressions F and A, the string formed by the character `'('`, followed by F, followed by the character `' '` (space), followed by A, followed by the character `')'` ((`(`F A`)`) is an expressionn (application).

Spent variables. The set of spent variables $VG(E)$ of an expression E is the set of all the variables from `'a'` to `'z'` that appear in the expression E, in any way. Equally,

$$VG(E) = \begin{cases} \{v\} & \textbf{If } E = v \\ VG(B) \cup \{v\} & \textbf{If } E = \backslash v\,.\,B \\ VG(F) \cup VG(A) & \textbf{If } E = \texttt{(}F\ \ A\texttt{)} \end{cases}$$

Free variables. The set of the free variables $VL(E)$ of an expression E is defined:

$$VL(E) = \begin{cases} \{v\} & \textbf{If } E = v \\ VL(B) - \{v\} & \textbf{If } E = \backslash v\,.\,B \\ VL(F) \cup VL(A) & \textbf{If } E = \texttt{(}F\ \ A\texttt{)} \end{cases}$$

Substitution. The substitution $E[x := E']$ of a variable v in an expression E by other expression E' is defined:

$$E[x := E'] = \begin{cases} v & \textbf{If } E \text{ is a variable } v \text{ different from } x \\ E' & \textbf{If } E = x \\ \backslash v\,.\,B & \textbf{If } E = \backslash v\,.\,B \text{ y } v = x \\ \backslash v\,.\,B[x := E'] & \textbf{If } E = \backslash v\,.\,B, v \neq x \text{ y } v \notin VL(E') \\ \texttt{(}F[x := E']\ \ A[x := E']\texttt{)} & \textbf{If } E = \texttt{(}F\ \ A\texttt{)} \end{cases}$$

Notice that, when E has the form $\backslash v\,.\,B$, it is not possible to do the substitution. This can always be corrected applying an alpha conversion, described in the next lines, in the expression $\backslash v\,.\,B$ replacing v with a no spent variable. *However, the inputs of the problems will be in a way that you will never need to do an alpha conversion to do a substitution.*

Input

A test data is a sequence of calculations, each one of them takes a line. The calculations can be "`G` E", "`L` E" or "`S` x E E'", where E and E' are valid expressions and x is a variable. No line will take more than 2000 characteres. In the calculation of types `S` we assure you that the substitution can be done it, without being necessary that you do any alpha conversion in any moment.

Output

For each calculation, your program must print a line with the answer. It must print the spent variables $VG(E)$ of E if the petition is of the type `G`, the free variables $VL(E)$ of E if is of the type `L`, and the substitution $E[x := E']$ if the petition is of the type `S`. It must print a set of variables as a sequence in alphabetical order (look the instances). Notice that the result of the substitution may be longer than 2000 characters.

Scoring

- **TestA:** **25 Points**

  Tests with less than 200 expressions that only contain calculation petitions of type `G`.

- **TestA:** **40 Points**

  Tests with less than 200 expressions that only contain calculation petitions of type `L`.

- **TestA:** **35 Points**

  Tests with less than 200 expressions that only contain calculation petitions of type `S`.

Sample input 1

```
G x
G \x.(x \y.(x y))
G (x y)
G \x.(x y)
G (\x.x f)
G (\x.(x \x.x) f)
G (\y.\x.y x)
G (((\c.\t.\e.((c t) e) \a.\b.a) a) b)
```

Sample output 1

```
x
xy
xy
xy
fx
fx
xy
abcet
```

Sample input 2

```
L x
L \x.(x \y.(x y))
L (x y)
L \x.(x y)
L (\x.x f)
L (\x.(x \x.x) f)
L (\y.\x.y x)
L (((\c.\t.\e.((c t) e) \a.\b.a) a) b)
```

Sample output 2

```
x

xy
y
f
f
x
ab
```

Sample input 3

```
S x x y
S y y y
S x x (x z)
S x (x x) (x z)
S x (x (y x)) (\x.(f f) g)
S x \x.x (a a)
S x \y.x (a a)
S x \y.(x \x.(x x)) (a z)
```

Sample output 3

```
y
y
(x z)
```

```
((x z) (x z))
((\x.(f f) g) (y (\x.(f f) g)))
\x.x
\y.(a a)
\y.((a z) \x.(x x))
```

Observation (not related to the problem)

If you are able to solve this problem, you are very close to be able to *evaluate* lambda expressions. We describe the necessary steps below.

Alpha conversion. The change in the name of the variable v in expressions like $\backslash v\,.\,B$ creates equivalent expressions.

Beta reductions. Any expression like $(\backslash x\,.\,B\ \ A)$ is reduced to $B[x := A]$, where may have been necessary to apply an alpha conversion to $\|x\,.\,B$ renaming x to no spent variable.

Normal form. It is said that an expression is in a normal form when no more beta reductions can be applied to any of its subexpressions.

Problem information

Author : Ángel Herranz
Translator : Carlos Molina
Generation : 2014-01-29 12:57:14