```
DOMINIOS RECURSIVOS

El dominio Ω que vimos, satisface el siguiente isomorfismo:

(*)     Ω ≈ (Σ' + Z x Ω)_⊥

φ ∈ Ω -> (Σ' + Z x Ω)_⊥
ψ ∈ (Σ' + Z x Ω)_⊥ -> Ω

φ y ψ continuas tales que:

      ⎧⊥                si x = <>
φ x = ⎨L0 y                si x = <y> con y ∈ Σ'
      ⎩L1 <n, w>            si x = <n> ++ w

      ⎧<>                   si x = ⊥
ψ x = ⎨<y>                  si x = L0 y
      ⎩<n> ++ w             si x = L1 <n, w>

φ y ψ son morfismos (preservan todo) y son inversas mutuas. Son isomorfismos.

Notación:

    Lnorm
Σ  ------->       L0
             Σ' ------->
Σ  ------->                           Lbottom                     ψ
    Labnorm              Σ' + Z x Ω ---------> (Σ' + Z x Ω)_⊥ -------> Ω
                  L1
            ZxΩ ------->

Lterm = ψ . Lbottom . L0 . Lnorm ∈ Σ -> Ω
Labort = ψ . Lbottom . L0 . Labnorm ∈ Σ -> Ω
Lout = ψ . Lbottom . L1 ∈ Z x Ω -> Ω
⊥_Ω = ψ(⊥) ∈ Ω

Ecuaciones:

[[skip]]s = Lterm s
[[fail]]s = Labort s
[[v:= e]]s = Lterm [s | v : [[e]]s]
                               ⎧[[c0]]s         si [[b]]s
[[if b then c0 else c1]]s = ⎨
                               ⎩[[c1]]s         c.c.
[[c0;c1]]s = [[c1]]_* ([[c0]]s)
donde, dada una f ∈ Σ -> Ω, denotamos por f_* la siguiente extensión de f a Ω. Así, f_* ∈ Ω -> Ω continua
definida por:

        ⎧⊥_Ω               si x = ⊥_Ω
f_* x = ⎨f s                si x = Lterm s
        ⎪Labort s           si x = Labort s
        ⎩Lout <n,f_* w>     si x = Lout <n,w>

En efecto, la clase pasada definimos así:

        ⎧<n1,..,nk> ++ f s        si x = <n1,..,nk,s>
f_* x = ⎨
        ⎩x                        c.c.

o

        ⎧<n1,..,nk>               si x = <n1,..,nk>
f_* x = ⎨<n1,..,nk> ++ f s        si x = <n1,..,nk,s>
        ⎪<n1,..,nk,<abort,s>>     si x = <n1,..,nk,<abort,s>>
        ⎩<n1,..,nk,...>           si x = <n1,..,nk,...>
```

```
Esta también se puede escribir así, f_* continua tal que:

        ⎧<>                   si x = <>
f_* x = ⎨f s                  si x = <s>
        ⎪<<abort,s>>          si x = <<abort,s>>
        ⎩<n> ++ f_* w         si x = <n> ++ w

[[catchin c0 with c1]]s = [[c1]]_♣ ([[c0]]s)
donde, dada una f ∈ Σ -> Ω, denotamos por f_♣ la siguiente extensión de f a Ω. Así, f_♣ ∈ Ω -> Ω continua
definida por:


         ⎧⊥_Ω                  si x = ⊥_Ω
f_♣ x = ⎨Lterm s              si x = Lerm s
         ⎪f s                  si x = Labort s
         ⎩Lout <n,f_♣ w>       si x = Lout <n,w>

[[while b do c]] = sup'(F^i ⊥')

              ⎧w_* ([[c]]s)     si [[b]]s
donde F w s = ⎨
              ⎩Lterm s          c.c.

[[newvar v:= e in c]]s = (λs'∈Σ. [s' | v : s v])_† ([[c]][s | v : [[e]]s])
donde, dada una f ∈ Σ -> Σ, denotamos por f_† la siguiente extensión de f a Ω. Así, f_† ∈ Ω -> Ω continua
definida por:

        ⎧⊥_Ω                si x = ⊥_Ω
f_† x = ⎨Lterm (f s)        si x = Lterm s
        ⎪Labort (f s)       si x = Labort s
        ⎩Lout <n,f_† w>     si x = Lout <n,w>

[[!e]]s = Lout <[[e]]s, Lterm s>

INPUT

Agregamos el comando

<comm> ::= ?<var>

Todas las ecuaciones como (*) que usen +, x, ()_⊥ y -> tienen solución. Puede considerarse Ω definido
por (*). Para contemplar también el input, vamos a modificar dicha definición.

Agregamos al isomorfismo:

Ω ≈ (Σ' + Z x Ω + (Z -> Ω))_⊥

φ ∈ Ω -> (Σ' + Z x Ω + (Z -> Ω))_⊥
ψ ∈ (Σ' + Z x Ω + (Z -> Ω))_⊥ -> Ω

Notación:

    Lnorm
Σ  ------->       L0
             Σ' ------->
Σ  ------->
    Labnorm
                  L1                             Lbottom                                 ψ
            ZxΩ ------->    Σ' + Z x Ω + (Z -> Ω) ---------> (Σ' + Z x Ω + (Z -> Ω)_⊥ -------> Ω

                  L2
       (Z -> Ω) ------->



Lterm = ψ . Lbottom . L0 . Lnorm ∈ Σ -> Ω
Labort = ψ . Lbottom . L0 . Labnorm ∈ Σ -> Ω
```

```
Lout = ψ . Lbottom . L1 ∈ Z x Ω -> Ω
Lin = ψ . Lbottom . L2 ∈ (Z -> Ω) -> Ω
⊥_Ω = ψ(⊥) ∈ Ω

Ecuaciones:

[[skip]]s = Lterm s
[[fail]]s = Labort s
[[v:= e]]s = Lterm [s | v : [[e]]s]
                                 ⎧[[c0]]s         si [[b]]s
[[if b then c0 else c1]]s = ⎨
                                 ⎩[[c1]]s         c.c.
[[c0;c1]]s = [[c1]]_* ([[c0]]s)
donde, dada una f ∈ Σ -> Ω, denotamos por f_* la siguiente extensión de f a Ω. Así, f_* ∈ Ω -> Ω continua
definida por:

         ⎧⊥_Ω              si x = ⊥_Ω
         ⎪f s              si x = Lterm s
f_* x = ⎨Labort s         si x = Labort s
         ⎪Lout <n,f_* w>   si x = Lout <n,w>
         ⎩Lin (f_* . g)    si x = Lin g

[[catchin c0 with c1]]s = [[c1]]_♣ ([[c0]]s)
donde, dada una f ∈ Σ -> Ω, denotamos por f_♣ la siguiente extensión de f a Ω. Así, f_♣ ∈ Ω -> Ω continua
definida por:


          ⎧⊥_Ω                si x = ⊥_Ω
          ⎪Lterm s            si x = Lerm s
f_♣ x = ⎨f s                si x = Labort s
          ⎪Lout <n,f_♣ w>     si x = Lout <n,w>
          ⎩Lin (f_♣ . g)      si x = Lin g

[[while b do c]] = sup'(F^i ⊥')

                 ⎧w_* ([[c]]s)     si [[b]]s
donde F w s = ⎨
                 ⎩Lterm s          c.c.

[[newvar v:= e in c]]s = (λs'∈Σ. [s' | v : s v])_✝ ([[c]][s | v : [[e]]s])
donde, dada una f ∈ Σ -> Σ, denotamos por f_✝ la siguiente extensión de f a Ω. Así, f_✝ ∈ Ω -> Ω continua
definida por:

         ⎧⊥_Ω                si x = ⊥_Ω
         ⎪Lterm (f s)        si x = Lterm s
f_✝ x = ⎨Labort (f s)       si x = Labort s
         ⎪Lout <n,f_✝ w>     si x = Lout <n,w>
         ⎩Lin (f_✝ . g)      si x = Lin g

[[!e]]s = Lout <[[e]]s, Lterm s>
[[?v]]s = Lin (λn∈Z. Lterm [s | v : n])
```