Lenguajes de Programación - Tarea 2

Integrantes

Nombre	No. de cuenta
Cureño Sánchez Misael	418002485
González Mancera Ivette	316014490

Ejercicios

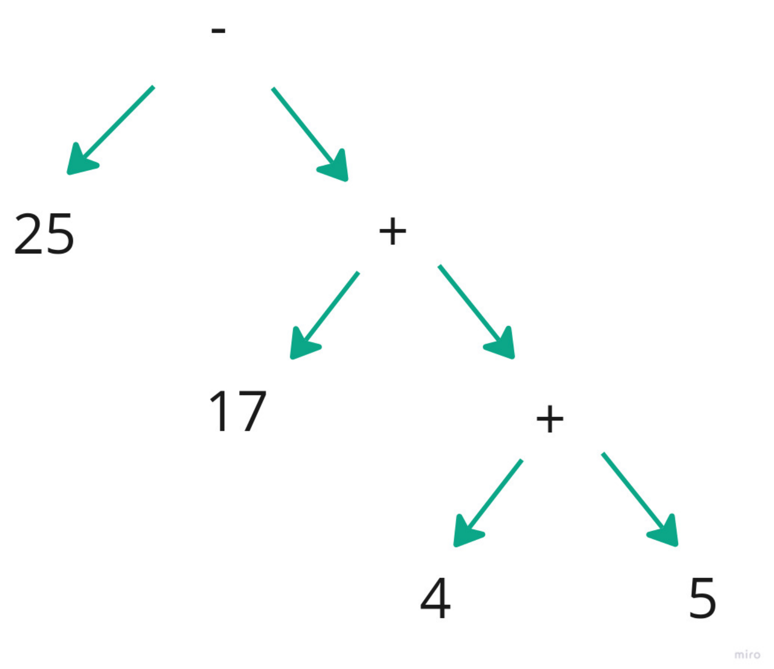
1. Dadas las siguientes expresiones en la gramática WAE dispuesta en sintaxis concreta, da la respectiva representación utilizando sintaxis abstracta por medio de los Árboles de Sintaxis Abstracta (ASA) correspondientes. En caso de no poder generar el árbol, justifica.

```
Tomando como referencia la siguiente construcción para ASA

;; TDA para representar los árboles de sintaxis
;; abstracta del lenguaje WAE .
(define -type WAE
    [id (id symbol ?)]
    [num (n numero ?)]
    [binop (f operador ?) (izq WAE ?) (der WAE ?)]
    [with (id symbol ?) (value WAE ?) (body WAE ?)])
```

- **a)**

```
{- 25 {+ 17 {+ 4 5}}}
```



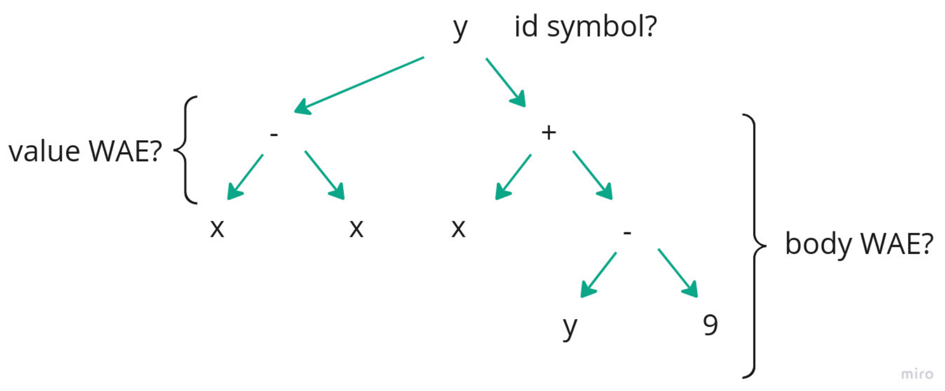
- **b)**

```
{- {+ 30 {+ 44}}}
```

No hay representación posible para {+ 44} en ASA, por lo tanto no se puede generar el árbol

- **c)**

```
{with {x 4}
    {with {y {- x x}}
        {+ x {- y 9}}}}
```



2. Dadas las siguientes expresiones en la gramática WAE dispuesta en sintaxis concreta, da la sintaxis abstracta correspondiente y realiza la sustitución que se indica.

   - **a)**

     `e = {+ a {+ b {- 32 57}}}`

     ```
     (parse e)

     (parse {+ a {+ b {- 32 57}}} )

     (add (parse a) (parse {+ b {- 32 57}} ))

     (add (id 'a) (add (parse b) (parse {- 32 57} )))

     (add (id 'a) (add (id 'b) (sub (parse 32) (parse 57))))

     (add (id 'a) (add (id 'b) (sub (num 32) (num 57))))
     ```

     `(subst (parse e) 'a (add (num 3) (num 4)))`

     ```
     (subst
         (parse e)
         'a
         (add (num 3) (num 4)))
     ```

```
(subst
    (add
        (id 'a)
        (add
            (id 'b)
            (sub (num 32) (num 57))))
    'a
    (add (num 3) (num 4)))
```

```
(add
    (add (num 3) (num 4))
    (subst
        (add
            (id 'b)
            (sub (num 32) (num 57)))
        'a
        (add (num 3) (num 4))))
```

```
(add
    (add (num 3) (num 4))
    (add
        (id 'b)
        (sub (num 32) (num 57))))
```

- **b)**

`e = {with {y {- 30 {- y z}}} {- 30 {+ y z}}}`

```
(parse e)

(parse {with {y {- 30 {- y z}}} {- 30 {+ y z}}})

(with
    (parse y)
    (parse {- 30 {- y z}})
    (parse {- 30 {+ y z}}))

(with
    (id 'y)
    (sub (parse 30) (parse {- y z}))
    (sub (parse 30) (parse {+ y z})))

(with
    (id 'y)
    (sub (num 30) (sub (parse y) (parse z)))
    (sub (num 30) (add (parse y) (parse z))))
```

```
(with
    (id 'y)
    (sub (num 30) (sub (id 'y) (id 'z)))
    (sub (num 30) (add (id 'y) (id 'z))))
```

`(subst (parse e) 'y (id 'w))`

```
(subst
    (with
        (id 'y)
        (sub
            (num 30)
            (sub (id 'y) (id 'z))
        )
        (sub
            (num 30)
            (add (id 'y) (id 'z))
        )
    )
    'y
    (id 'w)
)
```

Como en el primer paso se intenta sustituir en la declaracion del with, y el identificador coincide con el que estamos buscando sustituir, detenemos el algoritmo en este punto.

```
(with
    (id 'y)
    (sub
        (num 30)
        (sub (id 'y) (id 'z))
    )
    (sub
        (num 30)
        (add (id 'y) (id 'z))
    )
)
```

- **c)**

`e = {with {y {- 30 {- y z}}} {- 30 {+ y z}}}`

```
(parse e)
```

```
(parse {with {y {- 30 {- y z}}} {- 30 {+ y z}}} )

(with
    (parse y)
    (parse {- 30 {- y z}})
    (parse {- 30 {+ y z}}))
)

(with
    (id 'y)
    (sub (parse 30) (parse {- y z}))
    (sub (parse 30) (parse {+ y z}))
)

(with
    (id 'y)
    (sub (num 30) (sub (parse y) (parse z)))
    (sub (num 30) (add (parse y) (parse z)))
)

(with
    (id 'y)
    (sub (num 30) (sub (parse y) (parse z)))
    (sub (num 30) (add (parse y) (parse z)))
)

(with
    (id 'y)
    (sub (num 30) (sub (id 'y) (id 'z)))
    (sub (num 30) (add (id 'y) (id 'z)))
)
```

`(subst (parse e) 'z (id 'v))`

```
(subst
    (parse e)
    'z
    (id 'v))


(subst
    (with
        (id 'y)
        (sub (num 30) (sub (id 'y) (id 'z)))
        (sub (num 30) (add (id 'y) (id 'z)))
    )
    'z
    (id 'v))
```

```
(with
    (id 'y)
    (subst
        (sub (num 30) (sub (id 'y) (id 'z)))
        'z
        (id 'v))
    (subst
        (sub (num 30) (sub (id 'y) (id 'z)))
        'z
        (id 'v)))


(with
    (id 'y)
    (sub
        (subst (num 30) 'z (id 'v))
        (subst (sub (id 'y) (id 'z)) 'z (id 'v) ))
    (sub
        (subst (num 30) 'z (id 'v))
        (subst (add (id 'y) (id 'z)) 'z (id 'v) )))


(with
    (id 'y)
    (sub (num 30) (sub (id 'y) (id 'v)) )
    (sub (num 30) (add (id 'y) (id 'v)) ))


(sub
    (num 30)
    (add
        (sub (num 30) (sub (id 'y) (id 'v)))
        (id 'v)))
```

3. Sea la siguiente expresión definida usando lenguaje WAE. Da la sintaxis abstracta esta expresión. Muestra el proceso de evaluación mediante la función interp y responde las siguientes preguntas.

```
{with {a 3}
    {with {b 9}
        {with {c 4}
            {with {d 11}
                {+ a {+ b {+ c d}}}}}}}
```

Sintaxis abstracta:

```
(parse e)
```

```
(parse
    {with {a 3}
        {with {b 9}
            {with {c 4}
                {with {d 11}
                    {+ a {+ b {+ c d}}}}}}})

(with
    (parse a)
    (parse 3)
    (parse
        {with {b 9}
            {with {c 4}
                {with {d 11}
                    {+ a {+ b {+ c d}}}}}}))


(with (id 'a) (num 3)
    (with
        (parse b)
        (parse 9)
        (parse
            {with {c 4}
                {with {d 11}
                    {+ a {+ b {+ c d}}}}}})))


(with (id 'a) (num 3)
    (with (id 'b) (num 9)
        (with
            (parse c)
            (parse 4)
            (parse
                {with {d 11}
                    {+ a {+ b {+ c d}}}}}))))


(with (id 'a) (num 3)
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with
                (parse d)
                (parse 11)
                (parse
                    {+ a {+ b {+ c d}}})))))


(with (id 'a) (num 3)
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add
                    (parse a)
                    (parse {+ b {+ c d}}))))))
```

```
(with (id 'a) (num 3)
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add
                    (id 'a)
                    (add (parse b) (parse {+ c d})))))))


(with (id 'a) (num 3)
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add
                    (id 'a)
                    (add (id 'b) (add (parse c) (parse d))))))))


(with (id 'a) (num 3)
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add
                    (id 'a)
                    (add
                        (id 'b)
                        (add (id 'c) (id 'd))))))))
```

Evaluación:

```
(interp (parse e))

(interp
    (with (id 'a) (num 3)
        (with (id 'b) (num 9)
            (with (id 'c) (num 4)
                (with (id 'd) (num 11)
                    (add
                        (id 'a)
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd)))))))))


(interp
    (subst
        (with (id 'b) (num 9)
            (with (id 'c) (num 4)
                (with (id 'd) (num 11)
                    (add
                        (id 'a)
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd)))))))
        (id 'a)
        (interp (num 3))))


(interp
    (subst
        (with (id 'b) (num 9)
            (with (id 'c) (num 4)
                (with (id 'd) (num 11)
                    (add
                        (id 'a)
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd)))))))
        (id 'a)
        (num 3)))


(interp
    (with
        (id 'b)
        (subst (num 9) (id 'a) (num 3))
        (subst
            (with (id 'c) (num 4)
                (with (id 'd) (num 11)
                    (add
                        (id 'a)
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd))))))
            (id 'a)
            (num 3))))


(interp
    (with (id 'b) (num 9)
        (with
            (subst (id 'c) (id 'a) (num 3))
            (subst (num 4) (id 'a) (num 3))
            (subst
                (with (id 'd) (num 11)
                    (add
                        (id 'a)
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd)))))
                (id 'a)
                (num 3)))))
```

```
(interp
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with
                (subst (id 'd) (id 'a) (num 3))
                (subst (num 11) (id 'a) (num 3))
                (subst
                    (add
                        (id 'a)
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd))))
                    (id 'a)
                    (num 3))))))


(interp
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with
                (subst (id 'd) (id 'a) (num 3))
                (subst (num 11) (id 'a) (num 3))
                (subst
                    (add
                        (id 'a)
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd))))
                    (id 'a)
                    (num 3))))))


(interp
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add
                    (subst (id 'a) (id 'a) (num 3))
                    (subst
                        (add
                            (id 'b)
                            (add (id 'c) (id 'd)))
                        (id 'a)
                        (num 3)))))))


(interp
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add (num 3)
                    (add
                        (subst (id 'b) (id 'a) (num 3))
                        (subst
                            (add (id 'c) (id 'd))
                            (id 'a)
                            (num 3))))))))


(interp
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add (num 3)
                    (add (id 'b)
                        (add
                            (subst (id 'c) (id 'a) (num 3))
                            (subst (id 'd) (id 'a) (num 3)))))))))


(interp
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add (num 3)
                    (add (id 'b)
                        (add (id 'c) (id 'd))))))))


(with (id 'a) (num 3)
    (with (id 'b) (num 9)
        (with (id 'c) (num 4)
            (with (id 'd) (num 11)
                (add
                    (id 'a)
                    (add
                        (id 'b)
                        (add (id 'c) (id 'd))))))))
```

- ¿Cuántas veces se aplica el algoritmo de sustitución para evaluar esta expresión?

4. Convierte las siguientes expresiones usando la notación de índices de De Bruijn.

- **a)**

```
{with {v 2}
    {with {w 3}
        {with {x 4}
            {with {y {+ v {- w x} } }
                {with {z {with {v {+ w x}} v} }
                    {+ y {with {w {- y z}} {- w x}}}
                    }
                }
            }
        }
    }
```

Solución:

```
{with {2}
    {with {3}
        {with {4}
            {with {{+ <: 2 0 > {- <: 1 0 > <: 0 0 > } }}
                {with {{with { {+ <: 3 0>  <: 2 0> }} <: 0  1 > }}
                    {+ <: 2 0> {with {{- <: 3 0 > <: 1 0 > }} {- <: 0 0> <: 4 0 >
}}}
                }
            }
        }
    }
}
```

5. Dadas las siguientes expresiones representadas mediante índices de De Bruijn, obtén su respectiva versión usando los nombres de los identificadores de variables, iniciando por x, y, z, v, w.

   ```
   {with {1 2 3}
       {with {4 5 6}
           {with { {with {{+ <:0 1> <:1 2>} {- <:1 1> <:0 0>}}  3} }
               {with {<: 0 0>}
                   {+ <:3 2> {+ <:2 1> {+ <:1 0> <:0 0>}}}}}}}
   ```

   Procedimiento:

   ```
   {with {1 2 3}
       {with {4 5 6}
           {with
               {
                   {with { {+ <:0 1> <:1 2>} {- <:1 1> <:0 0>} } 3}
               }
               {with {<: 0 0>}
                   {+ <:3 2> {+ <:2 1> {+ <:1 0> <:0 0>}}}}}}}


   {with {1 {f 2} {d 3}}
       {with {{e 4} {c 5} 6}
           {with
               {
                   {b {with { {+ e d} {- f e} } 3}}
   ```

```
            }
            {with {a b}
                {+ d {+ c {+ b a}}}}}}}


{with {u 1}
    {with {f 2}
        {with {d 3}
            {with {e 4}
                {with {c 5}
                    {with {w 6}
                        {with {b {with { {+ e d} {- f e} } 3}}
                            {with {a b}
                                {+ d {+ c {+ b a}}}}}}}}}}}
```

Solución:

```
{with {x 1}
    {with {y 2}
        {with {z 3}
            {with {v 4}
                {with {w 5}
                    {with {u 6}
                        {with
                            {
                                b
                                {with {c {+ v z}}
                                    {with {d {- y v}}
                                        3
                                    }
                                }
                            }
                            {with {a b}
                                {+ z {+ w {+ b a}}}}}}}}}}}
```

6. Dada la siguiente expresión.

```
{with {w 2}
    {with {x 3}
        {with {y {+ w x}}
            {with { w -2}
                {with {x -3} {+ y y}}}}}}
```

- Evalúa la expresión. Muestra los pasos que se deben de hacer en cada una de sus derivaciones intermedias.

```
(interp (parse e))


(interp
    (parse
        {with {w 2}
            {with {x 3}
                {with {y {+ w x}}
                    {with { w -2}
                        {with {x -3} {+ y y}}}}}}))


(interp
    (with
        (parse w)
        (parse 2)
        (parse
            {with {x 3}
                {with {y {+ w x}}
                    {with { w -2}
                        {with {x -3} {+ y y}}}}})))


(interp
    (with (id 'w) (num 2)
        (with
            (parse x)
            (parse 3)
            (parse
                {with {y {+ w x}}
                    {with { w -2}
                        {with {x -3} {+ y y}}}}))))


(interp
    (with (id 'w) (num 2)
        (with (id 'x) (num 3)
            (with
                (parse y)
                (parse {+ w x})
                (parse
                    {with { w -2}
                        {with {x -3} {+ y y}}})))))


(interp
    (with (id 'w) (num 2)
        (with (id 'x) (num 3)
            (with (id 'y)
                (add (parse w) (parse x))
                (with
                    (parse w)
                    (parse -2)
                    (parse
                        {with {x -3} {+ y y}}))))))


(interp
    (with (id 'w) (num 2)
        (with (id 'x) (num 3)
            (with (id 'y) (add (id 'w) (id 'x))
                (with (id 'w) (num -2)
                    (with
                        (parse x)
                        (parse -3)
                        (parse {+ y y})))))))


(interp
    (with (id 'w) (num 2)
        (with (id 'x) (num 3)
            (with (id 'y) (add (id 'w) (id 'x))
                (with (id 'w) (num -2)
                    (with (id 'x) (num -3)
                        (add (parse y) (parse y))))))))


(interp
    (with (id 'w) (num 2)
        (with (id 'x) (num 3)
            (with (id 'y) (add (id 'w) (id 'x))
                (with (id 'w) (num -2)
                    (with (id 'x) (num -3)
                        (add (id 'y) (id 'y))))))))


(interp
    (subst
        (with (id 'x) (num 3)
            (with (id 'y) (add (id 'w) (id 'x))
                (with (id 'w) (num -2)
                    (with (id 'x) (num -3)
                        (add (id 'y) (id 'y))))))
        (id 'w)
        (interp (num 2))))


(interp
    (subst
        (with (id 'x) (num 3)
            (with (id 'y) (add (id 'w) (id 'x))
                (with (id 'w) (num -2)
                    (with (id 'x) (num -3)
                        (add (id 'y) (id 'y))))))
        (id 'w)
        (num 2)))


(interp
    (with (id 'x)
        (subst
            (num 3)
            (id 'w)
            (num 2))
        (subst
            (with (id 'y) (add (id 'w) (id 'x))
                (with (id 'w) (num -2)
                    (with (id 'x) (num -3)
                        (add (id 'y) (id 'y)))))
            (id 'w)
            (num 2))))


(interp
    (with (id 'x) (num 3)
        (subst
            (with (id 'y) (add (id 'w) (id 'x))
                (with (id 'w) (num -2)
                    (with (id 'x) (num -3)
                        (add (id 'y) (id 'y)))))
            (id 'w)
            (num 2))))


(interp
    (with (id 'x) (num 3)
        (with
            (id 'y)
            (subst
                (add (id 'w) (id 'x))
                (id 'w)
                (num 2))
            (subst
                (with (id 'w) (num -2)
                    (with (id 'x) (num -3)
                        (add (id 'y) (id 'y))))
                (id 'w)
                (num 2)))))


(interp
    (with (id 'x) (num 3)
        (with (id 'y)
            (add
                (subst
                    (id 'w)
                    (id 'w)
```

```
                     (num 2))
                 (subst
                     (id 'x)
                     (id 'w)
                     (num 2)))
             (with (id 'w) (num -2)
                 (with (id 'x)
                     (num -3)
                     (add (id 'y) (id 'y)))))))


(interp
    (with (id 'x) (num 3)
        (with (id 'y)
            (add (num 2) (id 'x))
            (with (id 'w) (num -2)
                (with (id 'x) (num -3)
                    (add (id 'y) (id 'y)))))))


(interp
    (subst
        (with (id 'y)
            (add (num 2) (id 'x))
            (with (id 'w) (num -2)
                (with (id 'x) (num -3)
                    (add (id 'y) (id 'y)))))
        (id 'x)
        (num 3)))


(interp
    (with (id 'y)
        (subst
            (add (num 2) (id 'x))
            (id 'x)
            (num 3))
        (subst
            (with (id 'w) (num -2)
                (with (id 'x) (num -3)
                    (add (id 'y) (id 'y))))
            (id 'x)
            (num 3))))


(interp
    (with (id 'y)
        (add
            (subst
                (num 2)
                (id 'x)
                (num 3))
            (subst
                (id 'x)
                (id 'x)
                (num 3)))
        (with (id 'w)
            (subst (num -2)
                (id 'x)
                (num 3))
            (subst
                (with (id 'x) (num -3)
                    (add (id 'y) (id 'y)))
                (id 'x)
                (num 3)))))


(interp
    (with (id 'y)
        (add (num 2) (num 3))
        (with (id 'w) (num -2)
            (with
                (id 'x)
                (subst
                    (num -3)
                    (id 'x)
                    (num 3))
                (subst
                    (add (id 'y) (id 'y))
                    (id 'x)
                    (num 3))))))


(interp
    (with (id 'y)
        (add (num 2) (num 3))
        (with (id 'w) (num -2)
            (with (id 'x) (num -3)
                (add
                    (subst (id 'y)
                        (id 'x)
                        (num 3))
                    (subst (id 'x)
                        (id 'x)
                        (num 3)))))))


(interp
    (with (id 'y)
        (add (num 2) (num 3))
        (with (id 'w) (num -2)
            (with (id 'x) (num -3)
                (add (id 'y) (num 3))))))


(interp
    (subst
        (with (id 'w) (num -2)
            (with (id 'x) (num -3)
                (add (id 'y) (num 3))))
        (id 'y)
        (add (num 2) (num 3))))


(interp
    (with
        (id 'x)
        (subst
            (num -3)
            (id 'y)
            (add (num 2) (num 3)))
        (subst
            (add (id 'y) (num 3))
            (id 'y)
            (add (num 2) (num 3)))))


(interp
    (with (id 'x) (num -3)
        (add
            (subst
                (id 'y)
                (id 'y)
                (add (num 2) (num 3)))
            (subst
                (num 3)
                (id 'y)
                (add (num 2) (num 3))))))


(interp
    (with (id 'x)
        (num -3)
        (add
            (add (num 2) (num 3))
            (num 3))))


(interp
    (subst
        (add
            (add (num 2) (num 3))
            (num 3))
        (id 'x)
        (num -3)))


(interp
    (add
        (subst
            (add (num 2) (num 3))
            (id 'x)
```

```
                (num -3))
            (subst
                (num 3)
                (id 'x)
                (num -3))))


(interp
    (add
        (add
            (subst
                (num 2)
                (id 'x)
                (num -3))
            (subst
                (num 3)
                (id 'x)
                (num -3)))
        (num 3)))


(interp
    (add
        (add (num 2) (num 3))
        (num 3)))


(num
    (+
        (num (interp (add (num 2) (num 3))))
        (num (interp (num 3)))))


(num
    (+
        (num
            (+
                (num-n (interp (num 2)))
                (num-n (interp (num 3)))))
        (num 3)))


(num (+ (num (+ (num 2) (num 3))) (num 3)))


(num (+ (num (num 5)) (num 3)))


(num (+ (num 5) (num 3)))


(num (num 8))
```

```
(num 8)


8
```

- ¿Pueden haber otros resultados? ¿Por qué?

  No, dado que el algoritmo de sustitucion se asegura de que no existan inconsistencias al momento de elegir que valores vamos a replazar en que lugar; además en cuanto a la interpretación, si lo pensamos como un automata, con su respectiva función de transición solo existe una transición para cada estado determinado del autómata, por lo que el resultado siempre es el mismo.

- ¿Cuál es el resultado correcto en dado caso de haber más de un posible resultado? ¿Por qué?

  Si hubiesemos utilizado la estrategia de ambientes para la evaluación, podríamos haber obtenido un lenguaje con alcance estático o dinámico, pero como hicimos uso del algoritmo de sustitucion obtenemos siempre el alcance estatico por la manera en que se ejecuta el mismo.