TAREA 1
LENGUAJES DE PROGRAMACIÓN
MEDINA PEÑA RAÚL

1. Problema I

Hemos visto en clase que la definición de sustitución resulta en una operación ineficiente: en el peor caso es de orden cuadrático en relación al tamaño del programa (considerando el tamaño del programa como el número de nodos en el árbol de sintaxis abstracta).

También se vio la alternativa de diferir la sustitución por medio ambientes. Sin embargo, implementar un ambiente usando un stack no parece ser mucho más eficiente.
Responde las siguientes preguntas.

- Provee un esquema para un programa que ilustre la no-linealidad de la implementación de ambientes basada en un stack. Explica brevemente porque su ejecución en tiempo no es lineal con respecto al tamaño de su entrada.

- Describe una estructura de datos para un ambiente que un intérprete de FWAE pueda usar para mejorar su complejidad.

- Muestra como usaría el intérprete esta nueva estructura de datos.

- Indica cual es la nueva complejidad del interprete (análisis del peor caso) y de forma informal pero rigurosa pruébalo.

RESPUESTAS

1. El programa que usaremos es:

```
{with {x 1}
    {with {y 4}
        {with {z 5}
            {with {g {fun {y} {- y x}}}
                {with {x 3}
                    {g 13}}}}}}
```

Entonces el stack nos quedaría de la siguiente forma:

x	3
g	{fun {y} {- y x}}
z	5
y	4
x	1

Donde al momento de hacer la aplicación, {g 13} se busca la definición de g en el stack, tardando dos búsquedas.

g = {fun {y} {- y x}}

entonces, ahora se busca el valor de x en el stack, pero este valor está en el tope del stack siendo 3, por lo que el tiempo es constante, quedando la expresión de la siguiente manera:

{g 13} → {{fun {y} {- y x}} 13} → {{fun {y} {- y 3}} 13} → {- 13 3} = **10**

Entonces tardo 3 búsquedas en el stack, y si consideramos que una búsqueda en el stack toma un 1 (tiempo constante).
Por lo tanto, la implementación de ambientes puede tener una ejecución que no es lineal respecto al tamaño de entrada del programa.

2. La estructura de datos que se propone no es muy diferente de la que se usa.
La mejora radica en que si se guarda una definición de una función entonces se guarda también todo el ambiente extendido hasta donde apareció la función.

Esto nos daría una mejora en el tiempo de búsqueda para las funciones, ya que no tendría que buscar en todo el stack, si no que al ver la función buscaría en su propio stack.

NombreFuncion	CuerpoFunción	Env ← Ambiente extendido actual

3. El intérprete tendría que devolver algo diferente a una función, tendría que devolver una tripleta de datos, que es el nombre de la función, el cuerpo de la función y su ambiente actual.

2. Problema II

Dada la siguiente expresión de FWAE:

```
{with {x 4}
        {with {f {fun {y} {+ x y}}}
                {with {x 5}
                        {f 10}}}}
```

debe evaluar a (num 14) usando alcance estático, mientras que usando alance dinámico se obtendrá (num 15),
Ahora Ben un agudo pero excéntrico estudiante dice que podemos seguir usando alcance dinámico mientras tomemos el valor más viejo de x en el ambiente en vez del nuevo y para este ejemplo él tiene razón.

- ¿Lo que dice Ben está bien en general? si es el caso justifícalo.

- Si Ben está equivocado entonces da un programa de contraejemplo y explica porque la estrategia de evaluación de Ben podría producir una respuesta incorrecta.

RESPUESTA

Lo que dice Ben **NO** está bien en general.

Un contraejemplo, basándonos en el ejemplo anterior sería:

```
{with {x 1}
        {with {x 4}
                {with {f {fun {y} {+ x y}}}
                        {with {x 5}
                                {f 10}}}}}
```

En el caso de usar alcance estático el resultado sería (num 14), ya que el valor de x lo alcanza hacia arriba es decir con x = 4.

Pero si usamos alcance dinámico, el ambiente quedaría de la siguiente manera:

env4 = ((x 5) (f { fun {y} {+ x y}}) (x 4) (x 1))

Donde al usar la estrategia de Ben, el valor más viejo de x es 1, por lo que el resultado de la evaluación de la expresión sería (num 11), dándonos un resultado diferente al que obtuvimos al usar alcance estático.

3. Problema III

Dada la siguiente expresión de FWAE con with multi-parametrico:

```
{with { {x 5} {adder {fun {x} {fun {y} {+ x y}}}} {z 3} }

    {with { {y 10} {add5 {adder x}} }

        {add5 {with { {x {+ 10 z}} {y {add5 0}} }

            {+ {+ y x} z}}}}}
```

- Da la forma Bruijn de la expresión anterior.
- Realiza la corrida de esta expresión, es decir escribe explícitamente cada una de las llamadas tanto para subst y interp, escribiendo además los resultados parciales en sintaxis concreta.

RESPUESTAS

1. Forma de Bruijn:

```
{with { 5 {fun{x} {fun{y} {+ x y}}} 3 }

    {with { 10 {<:01> <:00>} }

        {<:01> {with { {+ 10 <:12>} {<:01> 0}}

            {+ {+ <:01> <:00>} <:22>}}}}}
```

2. Realiza la corrida de la expresión:

> (interp {with ({x 5} {adder {fun {x} {fun {y} {+ x y}}}} {z 3}) (with {{y 10} {add5 {adder x}}} {add5 {with {{x {+ 10 z}} {y {add5 0}}} {+ {+ y x} z}}}})

>(interp
(subst (with {{y 10} {add5 {adder x}}} {add5 {with {{x {+ 10 z}} {y {add5 0}}} {+ {+ y x} z}}})})
({x 5} {adder {fun {x} {fun {y} {+ x y}}}} {z 3})
(interp (10 {adder x} {with {{x {+ 10 z}} {y {add5 0}})

Ahora haremos las llamadas a subst para cada una de los named-expr
Con
Para {y 10} → 10
Para {{add5 {adder x}} → {adder x} → x = 5 → {adder 5} → {add5 {fun{y} {+ 5 y}}}

Después se hace la aplicación de {add5 {with {{x {+ 10 z}} {y {add5 0}}} {+ {+ y x} z}}}}

De igual manera hacemos las sustituciones de los named-expr

Para {x {+ 10 z}} → z = 3 → { x {+ 10 3}} → {x 13}
Para {y {add5 0}} → {add5 0} → {+ 5 0} → {y 5}

Y ahora pasamos a evaluar {+ {+ y x} z}
Donde tomamos {+ {+ 5 13} 3} → 21

Y por último evaluamos la aplicación de {add5 21} → {+ 5 21} → 26

Por lo que el resultado es 26.