Tarea 1

Problema 1
~Provee un esquema para un programa que ilustre la no-linealidad de la implementación{on de ambientes basada en un stack. Explica brevemente por qué su ejecución en tiempo no es lineal con respecto al tamaño de su entrada.

```
{with {x 6}
        {with {y 7}
                {with {z 8}
                        {+ x {+ y z}}}}}

= {with {y 7}
        {with {z 8}
                {+ 6 {+ y z}}}}

= {with {z 8}
        {+ 6 {+ 7 z}}}

= {+ 6 {+ 7 8}}}


=21
```

La ejecución en tiempo no es lineal con respecto al tamaño de la entrada ya que en el ejemplo anterior el intérprete tiene que aplicar sustitución 3 veces;una por cada with.Por lo que se puede generalizar entonces si nuestro programa tuviese tamaño N(Medido por los nodos del àrbol de sintaxis abstracta)entonces cada substitución recorre el resto del programa al menos una vez.
Haciendo la complejidad de este intérprete O(N^2).

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

Primero visualizamos como es un FWAE:
NotaciÛn BNF:

```
<FWAE> ::= <num>
        |{+ <FWAE> <FWAE>}
        |{with {<id> <FWAE>}<FWAE>}
        |<id>
        |{<id> <FWAE>}
```

Si queremos mejorar su complejidad de cuadrática a algo mas decente, podemos implementar el intérprete con Tablas de dispersión donde nuestra llave será la variable x, y o z
y el valor será por el que hemos de sustituir en los withs

~Muestra como usarÌa el interprete Está nueva estructura de datos.
<FWAE> ::= <num>
|{+ <FWAE> <FWAE>}
|{with {<id> <FWAE>}<FWAE>}
|<id>
|{<id> <FWAE>}

Usaremos el ejemplo del primer inciso

{with {x 6}
{with {y 7}
{with {z 8}
{+ x {+ y z}}}}}

el primer with se hace en tiempo lineal al igual que el segundo y el tercero por lo que {6+{+7}8}

~Indica cual es la nueva complejidad del interprete(an·lisis del peor caso)y de forma informal pero rigurosa pruebalo.

como la complejidad interna de la estructura de Datos de la HasTable es lineal en el mejor caso, en el peor caso se tiene

Como las hashtables tienen complejidad lineal en e mejor de los casos pero en el peor de los casos es log (n) por lo que nuestro ínter preste al estar implementado en esta hashtable también tiene complejidades en el peor caso log(n)

- Ejercicio 2:

a) Lo que dice Ben está bien, mientras antes de la definición de la función en este caso "fun", solo haya una
asignación a la variable en cuestión (en este caso x), en caso de que haya más asignaciones de x, todas antes de
"fun", no va a tomar la misma que el alcance estático tomaría, ya que el estatico toma la asignación de x en el
ambiente anterior, que no es necesariamente la más vieja.

b) {with (x 4)
   {with (x 6)
    {with (f (fun (y) (+ x y)))
     {with (x 5)
      {f 10}}}}}

   - En este caso, estático evalúa 16, dinámico evalúa 15 y DinamicBen (Sí lo acabo de bautizar) evalúa 14.

-Ejercicio 3

a) {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}}}}}

{with {5 {fun {x} {fun {y} {+ x y}}} 3}
     {with {10 <: 0 1>}
          {<: 0 1> {with {{+ 10 <: 1 2 >} {<: 0 1 > 0}}
                       {+ {+ <: 1 0 > <: 2 0 >} <: 2 2 >}}}}}

b)
(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
          (interp 5))) *   * (interp 5) = 5

(subst({with {{y 10} {add5 {adder x}}}
                {add5 {with {{x {+ 10 z}} {y {add5 0}}}
                           {+ {+ y x} z}}}}
          x
          5)

(if(symbol=? y x 5)
     (with y
          (subst 10 x 5) = 10
          (subst {add5 {with {{x {+ 10 z}} {y {add5 0}}}
                           {+ {+ y x} z}}} x 5)))

```
(subst {add5 {with {{x {+ 10 z}} {y {add5 0}}}
                        {+ {+ y x} z}}} x 5)
(app add5 (subst {with {{x {+ 10 z}} {y {add5 0}}}
                        {+ {+ y x} z}} x 5))
(subst {with {{x {+ 10 z}} {y {add5 0}}}
                        {+ {+ y x} z}} x 5)
if(symbol=? x x)
   (with x
        (subst {+ 10 z} x 5) = (add 10 z)
        {+ {+ y x} z})

        (subst {+ 10 z} x 5)
          (add (subst 10 x 5)
               (subst z x 5))

               (subst 10 x 5)= 10
               (subst z x 5)
                 if(symbol=? z x) = z
                      #(add 10 z)

    (with x
         (add 10 z)
         {+ {+ y x} z}) = {+ {+ y (add 10 z)} z}

#(app add5 {+ {+ y (add 10 z)} z})



(subst {with {{x {+ 10 z}} {y {add5 0}}}
                        {+ {+ y x} z}} x 5)
if(symbol=? y x)
   (with y
        (subst {add5 0} x 5) = (app add5 0)
        (subst {+ {+ y x} z} x 5)) = (add (add y 5) z)

        (subst {add5 0} x 5)
           (app add5 (subst 0 x 5)) = (app add5 0)
           (subst 0 x 5)= 0


        #(subst {+ {+ y x} z} x 5)
           (add (subst {+ y x} x 5)) = (add y 5)
              (subst z x 5)       =  z

        (subst {+ y x} x 5)
```

```
(add (subst y x 5) = (add y 5)
    (subst x x 5))

(subst y x 5) (if (symbol=? y x)
                  y)
(subst x x 5) (if (symbol=? x x)
                  5)

(subst z x 5) (if (symbol=? z x)
                  z)

#(add (add y 5) z)



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}}}})
        adder
        (interp {fun {x} {fun {y} {+ x y}}}))


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

(subst({with {{y 10} {add5 {adder x}}}
            {add5 {with {{x {+ 10 z}} {y {add5 0}}}
                    {+ {+ y x} z}}}})
        z
        (interp 3)) **(interp 3)= 3

(if (symbol=? y z)
   (with y
       (subst 10 z 3) = 10
       (subst {add5 {with {{x {+ 10 z}} {y {add5 0}}} =  (app add5 (add (add y (add 10 3)) 3))
                    {+ {+ y x} z}}} z 3)))

       (subst 10 z 3) = 10
```

(subst {add5 {with {{x {+ 10 z}} {y {add5 0}}}
{+ {+ y x} z}}} z 3)

(app add5 (subst {with {{x {+ 10 z}} {y {add5 0}}} = (app add5 (add (add y (add 10 3)) 3))
{+ {+ y x} z}} z 3))
(subst {with {{x {+ 10 z}} {y {add5 0}}} = (add (add y (add 10 3)) 3)
{+ {+ y x} z}} z 3)
* (if (symbol=? x z)
(with x
(subst {+ 10 z} z 3) = (add 10 3)
(subst {+ {+ y x} z} z 3))) = (add (add y x) 3)
* (add (add y (add 10 3)) 3)

(subst {+ 10 z} z 3) = (add 10 3)
(add (subst 10 z 3) = 10
(subst z z 3) = (if (symbol=? z z)
3)

(subst {+ {+ y x} z} z 3) = (add (add y x) 3)
(add (subst {+ y x} z 3) = (add y x)
(subst z z 3)) = 3

(subst {+ y x} z 3) = (add (subst y z 3) = y
(subst x z 3) = x

(subst z z 3)= (if (symbol=? z z)
3)
(with x
(add 10 3)
(add (add y x) 3)) = (add (add y (add 10 3)) 3)

(with y
(subst 10 z 3) = 10
(subst {add5 {with {{x {+ 10 z}} {y {add5 0}}} = (app add5 (add (add y (add 10 3)) 3))
{+ {+ y x} z}} z 3))

(with y
10
(app add5 (add (add y (add 10 3)) 3))) = (app add5 (add (add 10 (add 10 3)) 3))

(subst {add5 0} z 3)
(app add5 (subst 0 z 3))

```
   (app add5 0)

(subst {+ {+ y x} z} z 3)
     (add (subst {+ y x} z 3) = (add y x)
          (subst z z 3)) = 3
     (subst {+ y x} z 3) = (add (subst y z 3) = y
                                (subst x z 3) = x
                           (add y x)

     (subst z z 3)= (if (symbol=? z z)
                        3)
(add (add y x) 3)

(with y
     (app add5 0)
     (add (add y x) 3))

*(add (add (app add5 0) x) 3)


*=(if(symbol=? add5 z)
  (with add5
        (subst {adder x} z 3) = (app adder x)
        (subst {add5 {with {{x {+ 10 z}} {y {add5 0}}}
                           {+ {+ y x} z}}} z 3))) = (app add5 (add (add y (add 10 3)) 3))


(subst {adder x} z 3)
    (app adder (subst x z 3)) = (app adder x)
    (subst x z 3) = (if( symbol=? x z)
                        x)

(subst {add5 {with {{x {+ 10 z}} {y {add5 0}}}
                           {+ {+ y x} z}}} z 3)
    (app add5 (subst {with {{x {+ 10 z}} {y {add5 0}}}
                           {+ {+ y x} z}} z 3))
    (subst {with {{x {+ 10 z}} {y {add5 0}}}
                           {+ {+ y x} z}} z 3)
          (if (symbol=? x z)
                (with x
                     (subst {+ 10 z} z 3) = (add 10 3)
                     (subst {+ {+ y x} z} z 3))) (add (add y x) 3)

=(app add5 (add (add y (add 10 3)) 3))

(with add5
```

(subst {adder x} z 3) = (app adder x)
(subst {add5 {with {{x {+ 10 z}} {y {add5 0}}}
{+ {+ y x} z}}} z 3))) = (app add5 (add (add y (add 10 3)) 3))

*=(app (app adder x) (add (add y (add 10 3)) 3))