1. Explica los tres tipos de variables que hay en un lenguaje de programación y explica por qué la evaluación de las expresiones depende únicamente de las variables libres. Da un ejemplo usando la gramática del lenguaje MiniLisp visto en clase.
   Solución.

   - *Variables de ligado.* Aquellas que definen el alcance de un conjunto de variables con el mismo nombre, junto a su valor.
   - *Variables libres.* Aquellas que no estén dentro del alcance de una variable de ligado.
   - *Variables ligadas.* Aquellas que estén definidas dentro del alcance de una variable de ligado.

   La evaluación de las expresiones depende únicamente de las variables libres ya que no corresponden con las de ligado y son las que se les puede cambiar su valor.

   Veamos el siguiente ejemplo.
   (let (a (+ a 3))
   (+ a a)) [a:=4]

   = (let (a (+ a[a:=4] 3))
   (+ a a))

   = (let (a (+ 4 3))
   (+ a a))

   = (let (a 7)
   (+ a a))

2. Dada la siguiente expresión de MiniLisp que usa notación de índices de Bruijn:

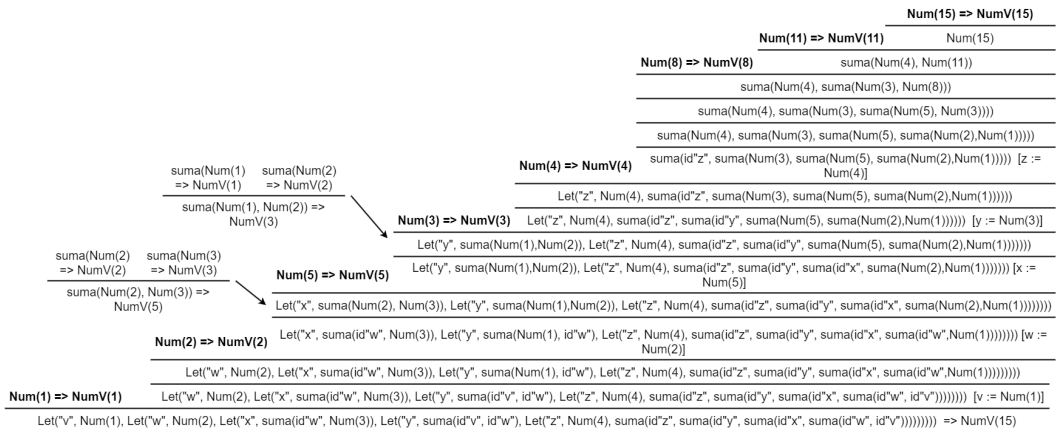
```
(let 1
   (let 2
      (let (+ <0> 3)
         (let (+ <2> <1>)
            (let 4
               (+ <0> (+ <1> (+ <2> (+ <3> <4>))))))))))
```

   1. Reconstruye la expresión usando variables.
      Solución.
      (let (v 1)
      (let (w 2)
      (let (x (+ w 3))
      (let (y (+ v w))
      (let (z 4)
      (+ z (+ y (+ x (+ w v)))))))))

   2. Señala las variables de ligado, ligadas y libres según corresponda.
      Solución.
      Donde
      Verde : Variables ligadas
      Rojo : Variables de ligado
      No hay variables libres.

3. Da la sintaxis abstracta de la expresión.
   Solución.
   Let("v", Num(1), Let("w", Num(2), Let("x", suma(id"w", Num(3)), Let("y", suma(id"v", id"w"), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(id"w", id"v")))))))))

4. Evalúa la expresión usando la semántica natural definida en clase.
   Solución.



5. Evalúa la expresión usando la semántica estructural definida en clase.
   Solución.

Let("v", Num(1), Let("w", Num(2), Let("x", suma(id"w", Num(3)), Let("y", suma(id"v", id"w"), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(id"w", id"v")))))))))
⟶Let("v", NumV(1), Let("w", Num(2), Let("x", suma(id"w", Num(3)), Let("y", suma(id"v", id"w"), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(id"w", id"v")))))))))
⟶Let("w", Num(2), Let("x", suma(id"w", Num(3)), Let("y", suma(NumV(1), id"w"), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(id"w",NumV(1))))))))))
⟶Let("w", NumV(2), Let("x", suma(id"w", Num(3)), Let("y", suma(NumV(1), id"w"), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(id"w",NumV(1))))))))))
⟶Let("x", suma(NumV(2), Num(3)), Let("y", suma(NumV(1),NumV(2)), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(NumV(2),NumV(1))))))))
⟶Let("x", suma(NumV(2), NumV(3)), Let("y", suma(NumV(1),NumV(2)), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(NumV(2),NumV(1))))))))
⟶Let("x", NumV(5), Let("y", suma(NumV(1),NumV(2)), Let("z", Num(4), suma(id"z", suma(id"y", suma(id"x", suma(NumV(2),NumV(1))))))))
⟶Let("y", suma(NumV(1),NumV(2)), Let("z", Num(4), suma(id"z", suma(id"y", suma(NumV(5), suma(NumV(2),NumV(1)))))))
⟶Let("y", NumV(3), Let("z", Num(4), suma(id"z", suma(id"y", suma(NumV(5), suma(NumV(2),NumV(1)))))))
⟶Let("z", Num(4), suma(id"z", suma(NumV(3), suma(NumV(5), suma(NumV(2),NumV(1))))))
⟶Let("z", NumV(4), suma(id"z", suma(NumV(3), suma(NumV(5), suma(NumV(2),NumV(1))))))
⟶suma(NumV(4), suma(NumV(3), suma(NumV(5), suma(NumV(2),NumV(1)))))
⟶suma(NumV(4), suma(NumV(3), suma(NumV(5), NumV(3))))
⟶suma(NumV(4), suma(NumV(3), NumV(8)))
⟶suma(NumV(4), NumV(11))
⟶NumV(15)

3. Chon Hacker desea extender al lenguaje MiniLisp con una instrucción para sumas arbitrarias, para lo cual agrega la siguiente descripción al manual de usuario:
   - Sintaxis: `(letsum (`x e_1`)` e_2 e_3`)`
   - Semántica: Para evaluar una expresión `letsum` debemos evaluar e_2 y e_3 obteniendo como resultados v_2 y v_3 respectivamente. El valor de la expresión `letsum` será entonces el valor de

$$\sum_{x=v_2}^{v_3} e_1$$

   - Observa que la variable x se considera ligada en la expresión e_1.
   - Ejemplos:
     La expresión $exp_1 =$ `(letsum (x x) (* 1 2) (+ 2 2))` se evalúa a $2+3+4=9$.
     La expresión $exp_2 =$ `(* 10 (letsum (z (* z z)) (+ 3 1) (- 8 2)))` se evalúa a $10\times(16+25+36)=770$.

1. Define la sintaxis concreta para la expresión **letsum**.
   Solución.
   $< exp > ::= < id >$
   $\quad | \ < NumV >$
   $\quad | \ (+ < exp >< exp >)$
   $\quad | \ (- < exp >< exp >)$
   $\quad | \ (* < exp >< exp >)$
   $\quad | \ (/ < exp >< exp >)$
   $\quad | \ (letsum(< id >< exp >) < exp >< exp >)$

2. Define la sintaxis abstracta para la expresión **letsum** mediante el juicio ASA.
   Solución.

$$\frac{}{\text{id(a)} \ \ \text{ASA}} \qquad \frac{}{\text{NumV(n)} \ \ \text{ASA}}$$

$$\frac{\text{x ASA} \quad \text{y ASA}}{\text{suma(x,y)} \ \ \text{ASA}} \qquad \frac{\text{x ASA} \quad \text{y ASA}}{\text{resta(x,y)} \ \ \text{ASA}}$$

$$\frac{\text{x ASA} \quad \text{y ASA}}{\text{mult(x,y)} \ \ \text{ASA}} \qquad \frac{\text{x ASA} \quad \text{y ASA}}{\text{div(x,y)} \ \ \text{ASA}}$$

$$\frac{\text{i ASA} \quad \text{v ASA} \quad \text{b1 ASA} \quad \text{b2 ASA}}{\text{letsum(i,v,b1,b2)} \ \ \text{ASA}}$$

Marín Parra José Guadalupe de Jesús

3. Define la semántica estructural para la expresión **letsum**.
Solución.

$$\frac{}{\text{id -> error}} \qquad \frac{}{\text{num(n) -> numV(n)}}$$

$$\frac{\text{i -> iv}}{\text{suma(i,d) -> suma(iv,d)}} \qquad \frac{\text{i -> iv}}{\text{resta(i,d) -> resta(iv,d)}} \qquad \frac{\text{i -> iv}}{\text{mult(i,d) -> mult(iv,d)}}$$

$$\frac{\text{d -> dv}}{\text{suma(i,d) -> suma(iv,dv)}} \qquad \frac{\text{d -> dv}}{\text{resta(i,d) -> resta(iv,dv)}} \qquad \frac{\text{d -> dv}}{\text{mult(i,d) -> mult(iv,dv)}}$$

$$\frac{}{\text{suma(iv,dv) -> iv + dv}} \qquad \frac{}{\text{resta(iv,dv) -> iv - dv}} \qquad \frac{}{\text{mult(iv,dv) -> iv * dv}}$$

$$\frac{\text{i -> iv}}{\text{div(i,d) -> div(iv,d)}} \qquad \frac{\text{b1 -> numV(b1)} \quad \text{b2 -> numV(b2)}}{\text{letsum(i,v,b1,b2) -> letsum(i,v,numV(b1),numV(b2))}}$$

$$\frac{\text{d -> dv}}{\text{div(i,d) -> div(iv,dv)}} \qquad \frac{\text{v[i := } \sum_{numV(b1)}^{numV(b2)} \text{] = vs}}{\text{letsum(i,v,numV(b1),numV(b2)) -> vs}}$$

$$\frac{}{\text{div(iv,dv) -> iv / dv}}$$

4. Da la sintaxis abstracta de exp_2 y evalúala usando tu semántica estructural.
Solución.

$$\frac{\text{numV(3) -> 3} \qquad \text{numV(1) -> 1}}{\text{suma(numV(3), numV(1)) -> suma(3,1)}}$$

$$\frac{}{\text{(mult(numV(10), (letsum(z, (mult(z, z)), suma(numV(3), numV(1)), resta(numV(8), numV(2))))))}}$$

$$\downarrow$$

$$\frac{\text{numV(8) -> 8} \qquad \text{numV(2) -> 2}}{\text{resta(numV(8), numV(2)) -> resta(8,2)}}$$

$$\frac{}{\text{(mult(numV(10), (letsum(z, (mult(z, z)), suma(3,1), resta(numV(8), numV(2))))))}}$$

$$\downarrow$$

$$\frac{\text{suma(3,1) -> 4} \qquad \text{resta(8,2) -> 6} \qquad \text{(mult(z, z)) -> } z^2}{\text{(mult(numV(10), (letsum(z, (mult(z, z)), suma(3,1), resta(8,2)))))}}$$

$$\downarrow$$

$$\frac{\text{letsum(z, } z^2\text{, 4, 6) -> 16 + 25 + 36 = 77} \qquad \text{numV(10) -> 10}}{\text{(mult(numV(10), (letsum(z, } z^2\text{, 4, 6))))}}$$

$$\downarrow$$

$$\frac{}{\text{(mult(10, 77)) -> 10*77 = 770}}$$