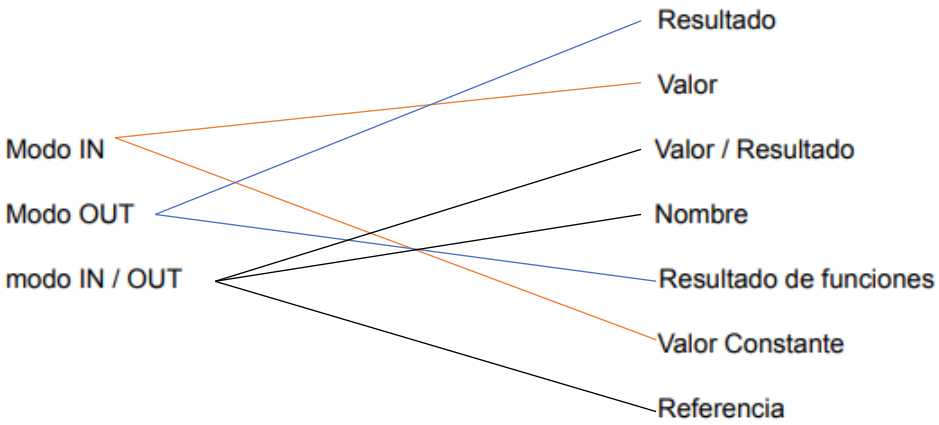
Práctica 6

Ejercicio 1:

a- Explique brevemente los siguientes conceptos

- **Parámetro:** es una forma de compartir datos entre diferentes unidades. Es la más flexible y permite la transferencia de diferentes datos en cada llamada. Proporciona ventajas en legibilidad y modificabilidad. Nos permiten compartir los datos en forma abstracta ya que indican con precisión qué es exactamente lo que se comparte
- **Parámetro real:** : es un valor u otra entidad utilizada para pasar a un procedimiento o función. Están en la parte de la invocación
- **Parámetro formal:** es una variable utilizada para recibir valores de entrada en una rutina, subrutina etc. Se ponen en la parte de la declaración. Es una variable local a su entorno.
- **Ligadura posicional:** los parámetros formales y reales se ligan según la posición en la llamada y en la declaración.
- **Ligadura por palabra clave o nombre:** los parámetros formales y reales se ligan por el nombre. Se debe conocer los nombres de los parámetros formales.

Ejercicio 2: Unir los siguientes puntos según corresponda y de una definición y un ejemplo de cada par.



Ejercicio 3:

a- Complete el siguiente cuadro según lo correspondiente a cada lenguaje:

Tipo de pasaje de parámetros	Lenguaje
• Por defecto con copia IN • Por resultado OUT • IN OUT	ADA

○ Para los tipos primitivos indica por valor-resultado ○ Para los tipos no primitivos, datos compuestos (arreglo , registro) se hace por referencia.	
• Por valor (Si se necesita por referencia se usa punteros) • Permite pasaje por valor constante, agregando const	C
• Por valor, pero si se pasa un objeto "mutable", no se hace una copia sino que se trabaja sobre él.	Ruby
• El único mecanismo contemplado es el paso por copia de valor. Pero como las variables de tipo no primitivos son todas referencias a variables anónimas en la heap, el paso por valor de una de estas variables son en realidad un paso por referencia de las variables	JAVA
• Se puede pasar de dos formas: ○ Inmutables: actuara como por valor ○ Mutables: No se hace una copia sino que se trabaja sobre él.	Python

b- Ada es más seguro que Pascal, respecto al pasaje de parámetros en las funciones. Explique por qué.

Ada es más seguro que Pascal en cuanto al pasaje de parámetros en las funciones debido a su sistema de tipos más estricto y su sintaxis más clara y explícita para especificar los modos de paso de parámetros.

c- Explique cómo maneja Ada los tipos de parámetros in-out de acuerdo al tipo de dato

Ada maneja los parámetros in-out de acuerdo al tipo de dato que se está utilizando, utilizando una técnica de paso por referencia para tipos de datos complejos, y una técnica de copia y devolución para tipos de datos más simples. Esto permite un manejo seguro y eficiente de los parámetros in-out en los programas Ada.

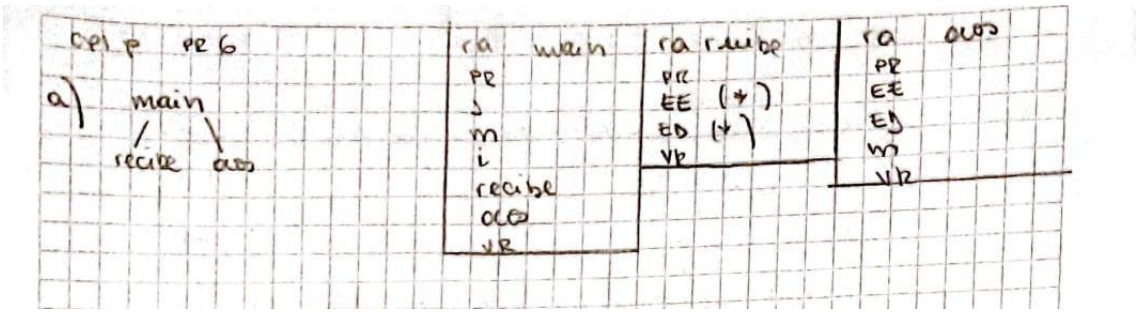
Ejercicio 4: Sea el siguiente programa escrito en Pascal-like

```
Procedure Main;
var j, m, i: integer;

Procedure Recibe (x:integer; y:integer);
        begin
                m:= m + 1 + y;
                x:=i + x + j;
                y:=m - 1;
                write (x, y, i, j, m);
        end;
```

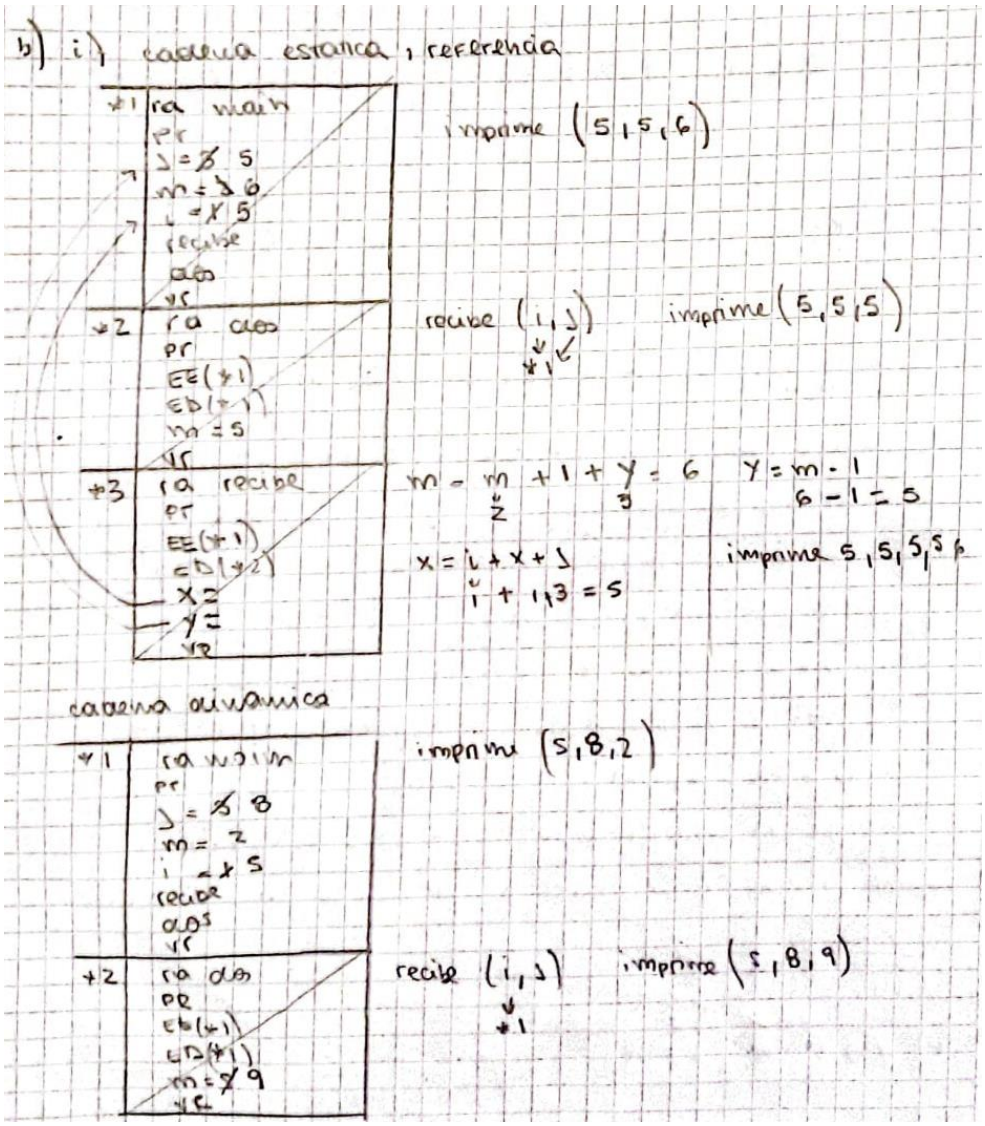
```
Procedure Dos;
        var m:integer;
        begin
                m:= 5;
                Recibe(i, j);
                write (i, j, m);
        end;
begin
        m:= 2;
        i:=1; j:=3;
        Dos;
        write (i, j, m);
end.
```

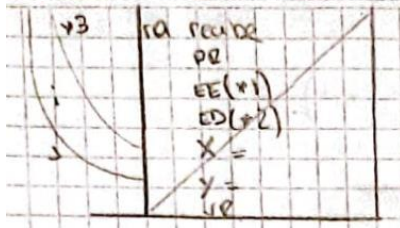
a- Arme el árbol de anidamiento sintáctico y el registro de activación de cada una de las unidades.



b- Decir qué imprime el programa suponiendo que para todas las variables que se pasan el pasaje de parámetros es por: (Deberá hacer la pila estática y dinámica para cada caso)

i- Referencia. ii- Valor. iii- Valor Resultado. iv- Nombre. v- Resultado.





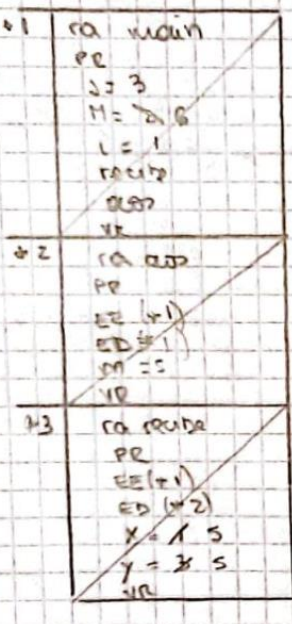
$m = \underset{\downarrow}{m} + 1 + \underset{\downarrow}{Y} = 9$ (m: 5, Y: 3)

$Y = \underset{\downarrow}{m} - 1 = 8$ (m: 9)

$X = 1 + \underset{\downarrow}{X} + 3 = 5$ (X: 1)

imprime (5, 8, 5, 8, 9)

ii) cadena estatica, por valor



imprime (1, 3, 6)

imprime (1, 3, 5)

$m = m + 1 + Y = 6$ (m: 2, Y: 3)

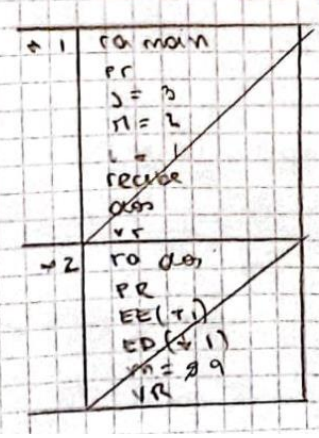
$Y = m - 1$

$6 - 1 = 5$

$X = 1 + X + 3 = 5$ (X: 1)

imprime (3, 1, 3, 6)

cadena dinamica



imprime (1, 3, 2)

recibe (1, 3)

imprime (1, 3, 9)

#3	ra recibe PR EE(#1) ED(#2) X = X 5 Y = X 8 VR

m x

m = 5 + 1 + 3 = 9

x = i + X + j = 5
1 + 1 + 3

y = m - 1 = 8
9 - 1

imprime (5, 8, 1, 3, 9)

iii) cadena estatica, valor resultado

#1	ra main PR i = X 5 M = X 6 i = X 5 recibe dos VR

imprime (5, 5, 6)

#2	ra dos PR EE (#1) ED (#1) m = 5 VR 5, 5

recibe (i, j) imprime (5, 5, 5)
1 8

#3	ra recibe PR EE(#1) ED(#2) X = (VR:i) = X 5 Y = (VR:j) = X 8 VR

m = m + 1 + Y = 6
2 + 1 + 3

X = i + X + j = 5
1 + 1 + 3

Y = m - 1 = 5
6 - 1

imprime (5, 8, 1, 3, 6)

cadena dinamica

#1	ra main PR j = X 8 M = 2 i = X 5 recibe dos VR

imprime (5, 8, 2)

#2	ra dos PR EE(#1) ED(#1) m = X 9 VR 5, 8

recibe (i, j) imprime (5, 8, 9)
1 3

#3	ra recibe PR EE(#1 ED(#2) X=(VR:i) = X 5 Y=(VR:j) = X 8 VR

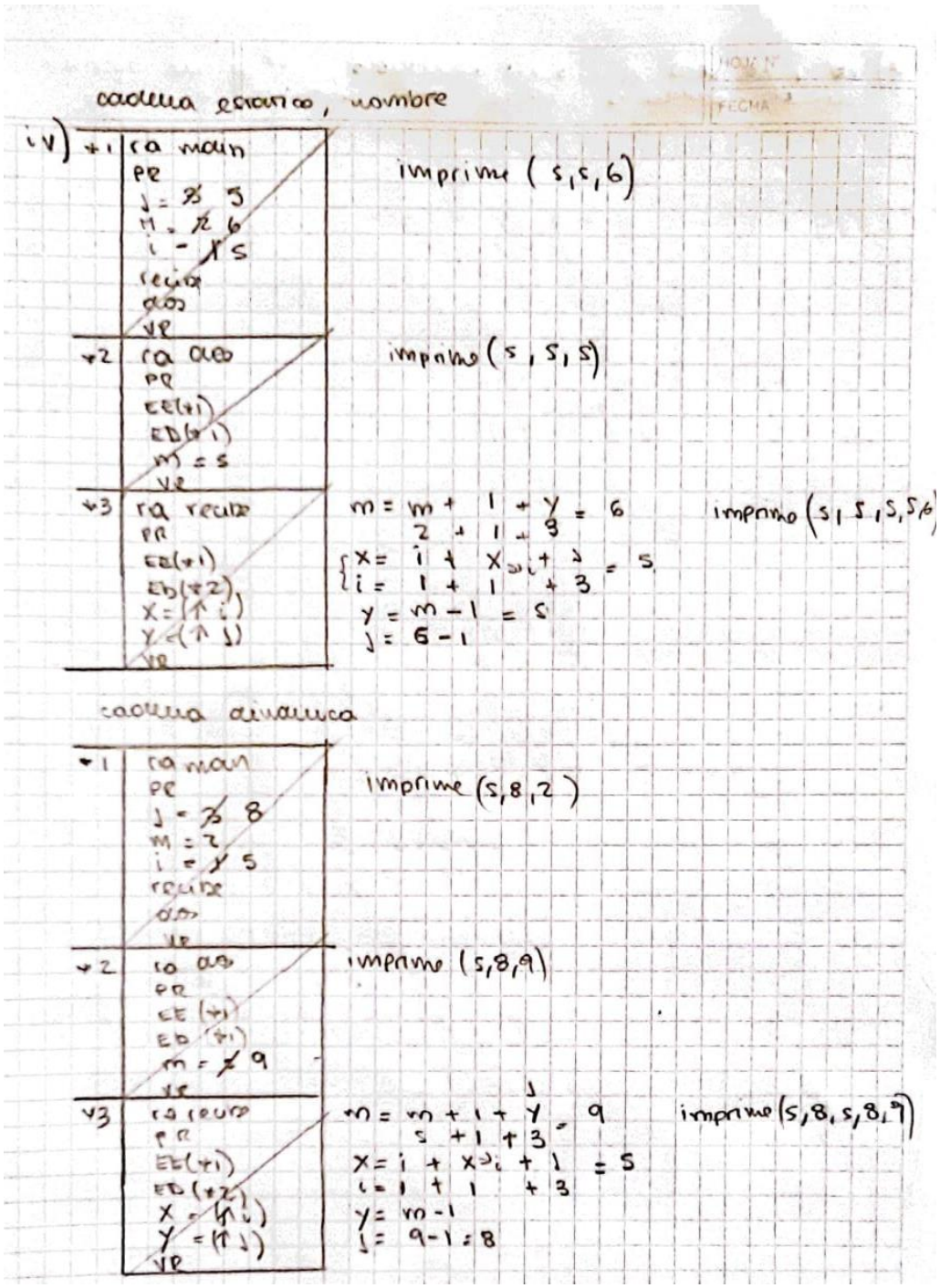
m = m + 1 + Y = 9
5 + 1 + 3

X = i + X + j = 5
1 + 1 + 3

Y = m - 1 = 8
9 - 1

imprime (5, 8, 1, 3, 9)

NOTA



c- ¿Existió algún caso que no pudo realizarlo porque saltó algún tipo de error? Diga cuál y por qué.

Si, en la cadena estática y dinámica resultado hay error en m = m + 1 + y ya que y no esta inicializada

d- ¿Dará el mismo resultado si se trata de un lenguaje que sigue la cadena dinámica? Justifique la respuesta realizando las pilas de activación

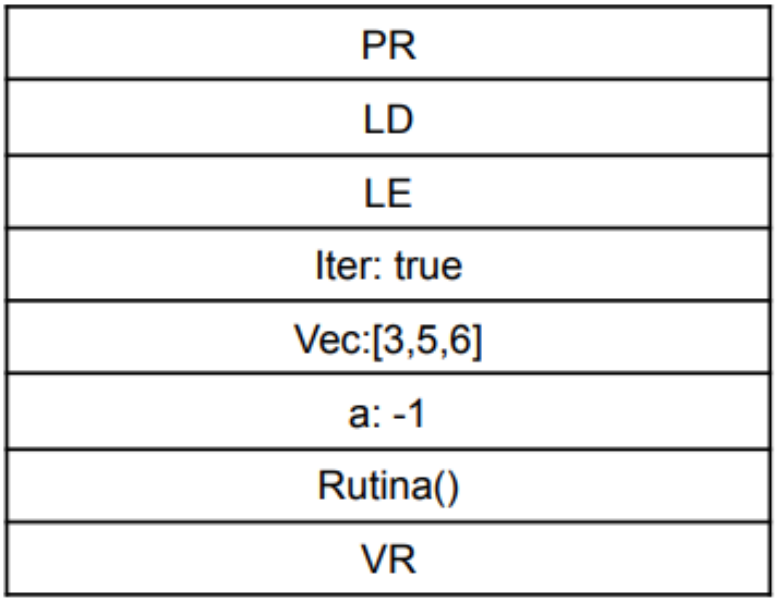
No, no será el mismo resultado, se puede observar en el punto b como no tienen los mismos resultados e impresiones.

Ejercicio 5: Suponiendo que se está ejecutando un programa con el siguiente registro de activación en memoria y se llama al procedimiento rutina(iter,vec,a). Determine el tipo de parámetro que se deben utilizar en el llamado para que los resultados sean los siguientes:

a) (4,6,7),(4,6,7), 2, 2

b) (3,5,6),(4,6,7), 2, 2

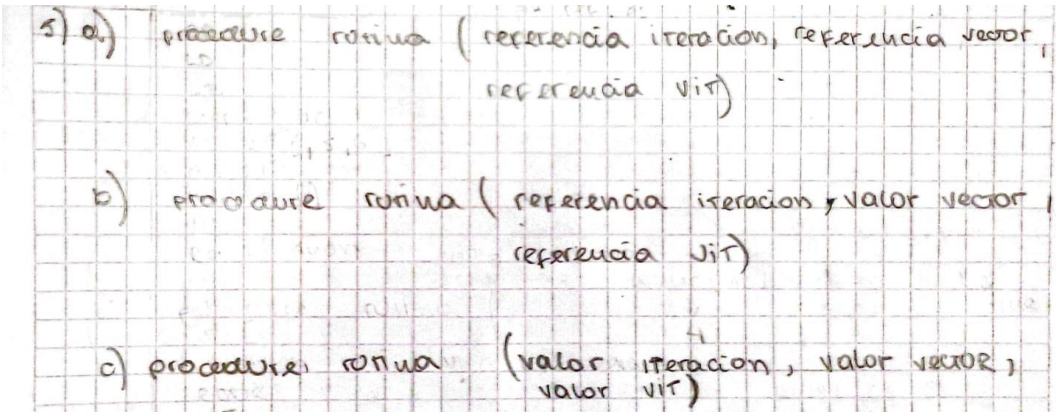
c) (3,5,6),(5,5,6), 0, -1




```
.....
procedura rutina(tipoParam iteracion,tipoParam vector,tipoParam vit):

        while iteracion begin
                vit = a+1
                vector[vit] = vector[vit]+1
                iteracion = (vector[vit] mod 2)==0
        end
        print vec
        print vector
        print vit
        print a
.....

rutina(iter,vec,a)
```

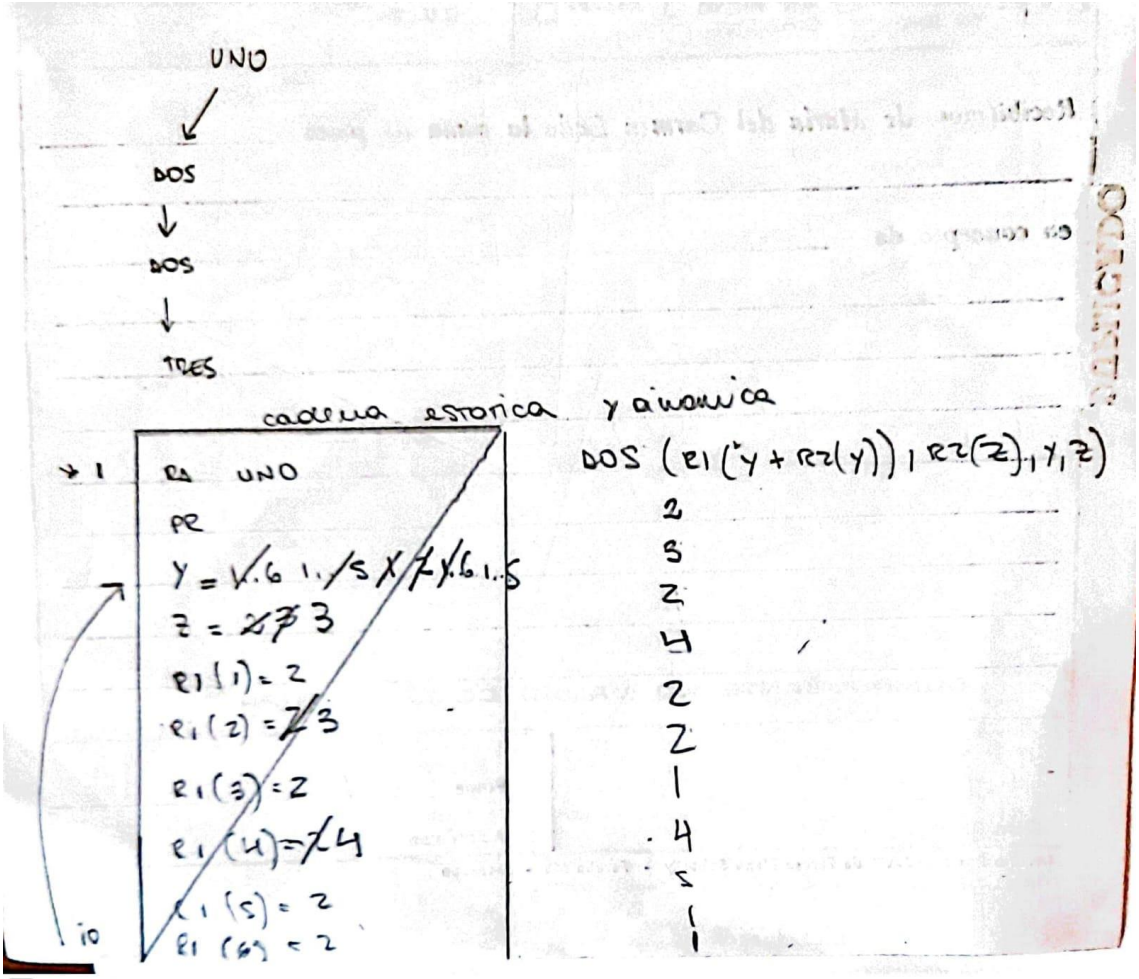


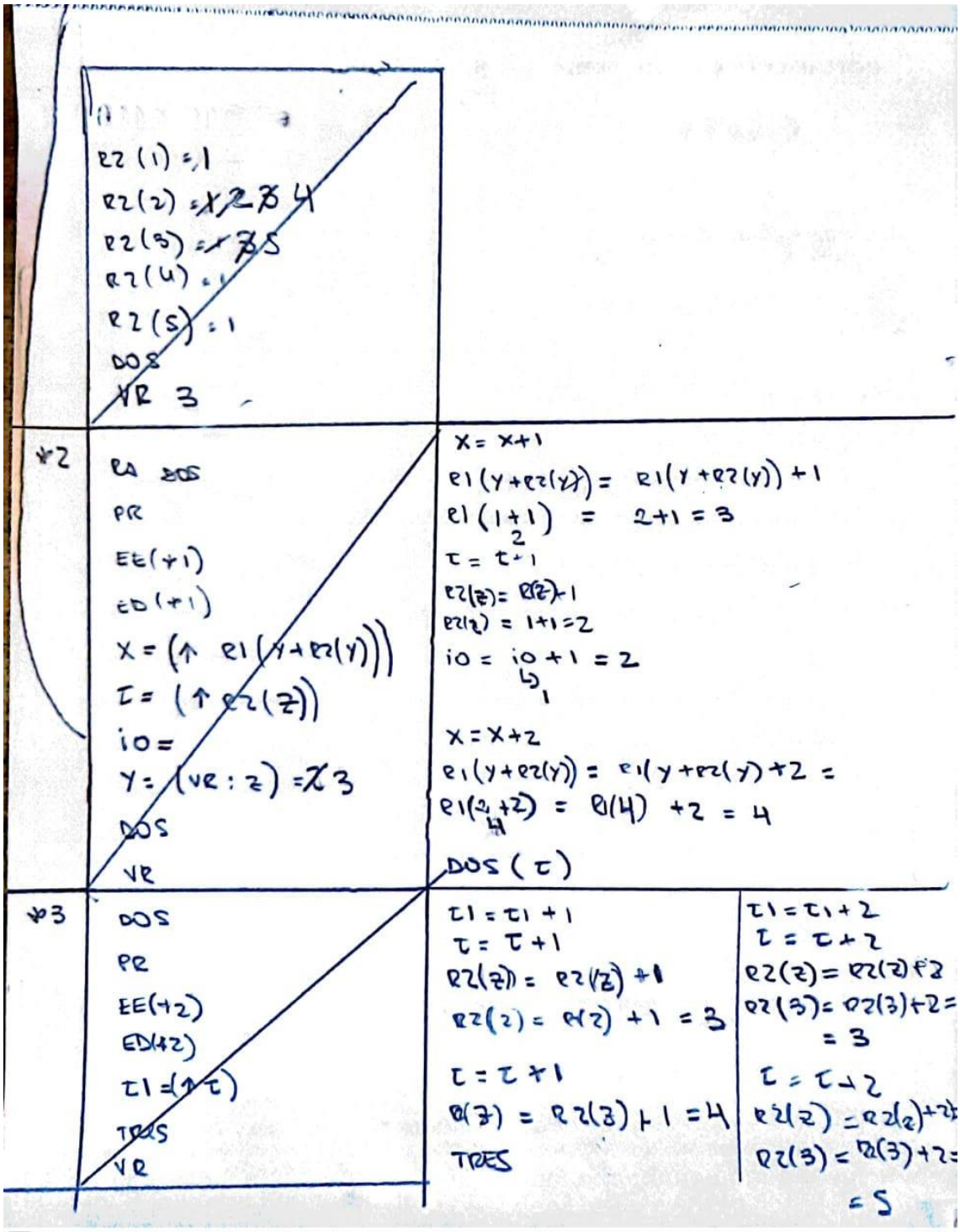
Ejercicio 6:Indique con un ejemplo el comportamiento del parámetro por nombre (en el parámetro formal) para los siguientes casos de parámetros reales:

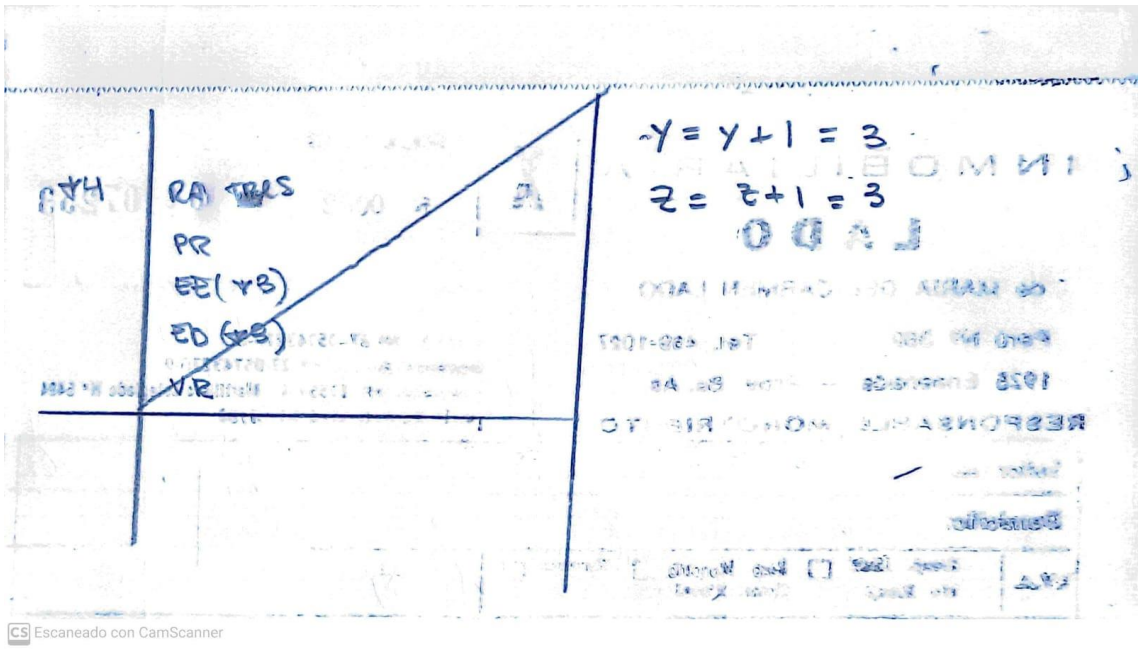
- **Un valor entero:** Un único valor se comporta exactamente igual que el pasaje por referencia
- **Una constante:** Si es una constante es equivalente a por valor
- **Un elemento de un arreglo:** Si es un elemento de un arreglo puede cambiar el suscripto entre las distintas referencias
- **Una expresión:** Si es una expresión se evalúa cada vez

Ejercicio 7: Realice la pila de ejecución del siguiente programa: a) siguiendo la cadena estática b)siguiendo la cadena dinámica

```
Procedure Uno;
y, z: integer;
r1:array[1..6] of integer;
r2:array[1..5] of integer;
        Procedure Dos( nombre x, t:integer;
var
                io:integer; valor-resultado
        y:integer);
                Procedure Dos( nombre
        t1:integer);
                        Procedure Tres;
                        begin
                        y:= y + 1;
                        z:= z + 1;
                        end;
                begin
                t1:= t1 + 1;
                t:= t + 1;
                Tres;
                t1:= t1 + 2;
                t:= t + 2;
                end;
        begin
        x:= x + 1;
        t:= t + 1;
        io:= io + 1;
        x:= x + 2;
        if z =2 then Dos ( t );
        end;
begin
for y:= 1 to 6 do r1(y):= 2;
for y:= 1 to 5 do r2(y):= 1;
z:= 2;
y:= 1;
Dos( r1( y + r2( y )), r2( z ), y, z);
for y:= 1 to 6 do write (r1(y));
for y:= 1 to 5 do write (r2(y));
end.
```







Ejercicio 8:

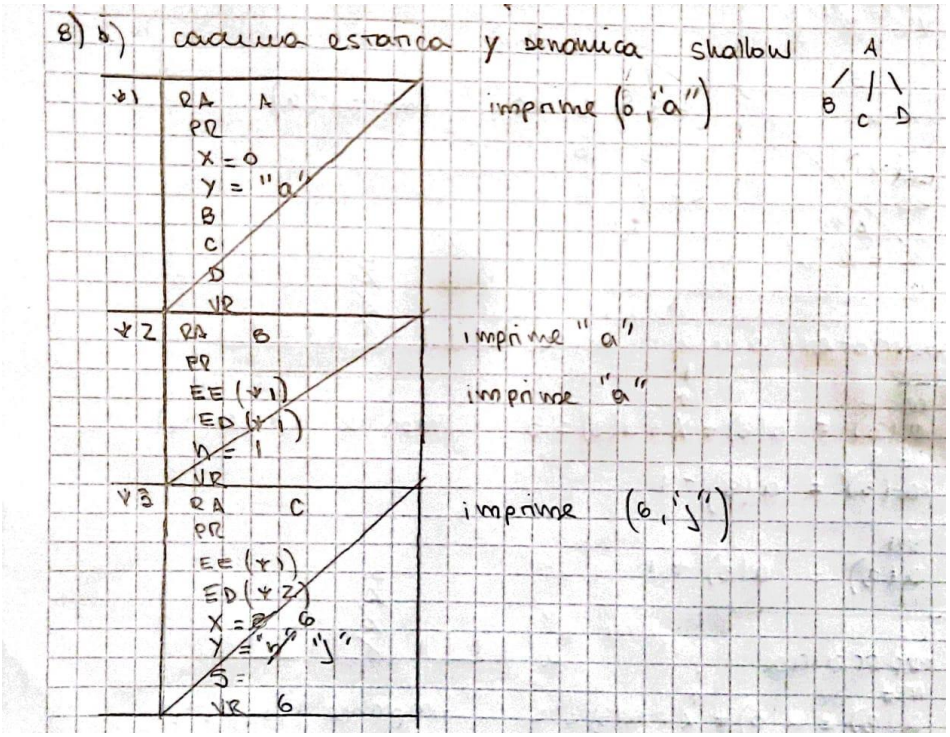
a) **Indique las diferencias entre los pasaje de subprogramas como parámetros deep y shallow.**

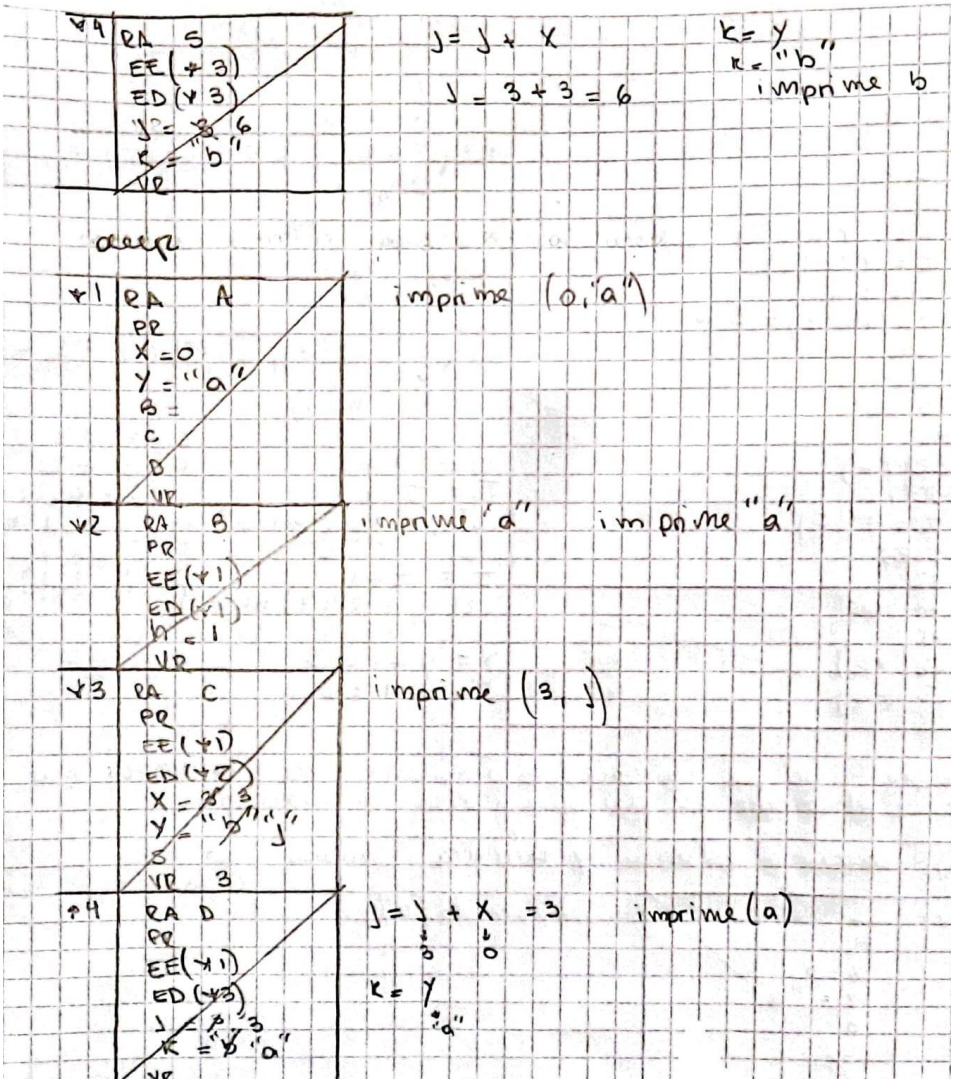
SHALLOW (parecido a buscar una variable por cadena dinámica): El ambiente de referencia, es el del subprograma que tiene el parámetro formal subprograma.
DEEP (parecido a buscar una variable por cadena estática): El ambiente es el del subprograma dónde está declarado el subprograma usado como parámetro real. Se utiliza en los lenguajes con alcance estático y estructura de bloque.

b) **Realice la pila estática y dinámica tanto con el pasaje de parámetros deep y shallow para el siguiente código.**

```
Program A
  Var x:integer;
  Var y: char;
  Procedure B;
     Var h:integer;
     Begin
       h:=1+x;
       Write (y);
       C(D);
       Write (y);
     End;
   Procedure C (Subrutina S);
      Var x:integer;
      Var y: char;
      Begin
        x:=3;
        y:= "b";
        x:=S(x,y)
        y:= "j";
        Write (x,y);
     End;
```

```
  Function D (j:integer, k:char);
     Begin
        j:=j+x;
        k:=y;
        Write (k);
        Return j;
      End;
BEGIN
  x:=0;
  y:="a";
  B();
  Write (x,y);
END.
```





Ejercicio 9: Sea el siguiente código escrito en Pascal like

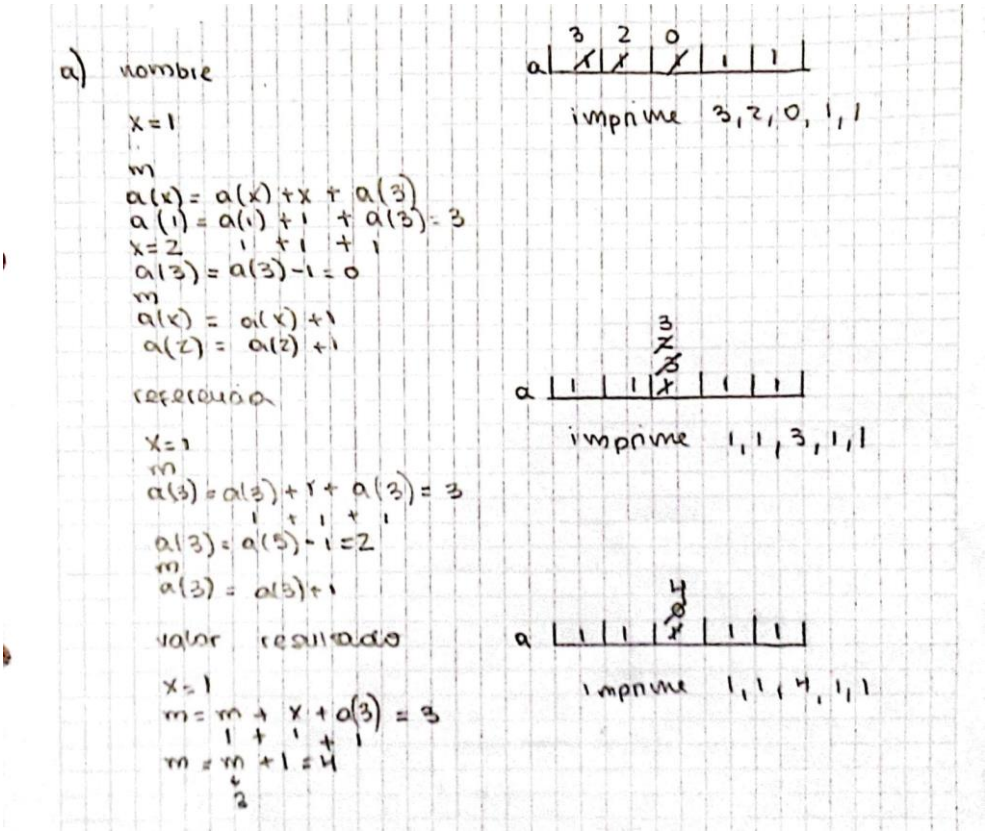
Procedure main
a: array(1..5) of integer;
x: integer;
i;integer;
Procedure Uno (tipo_pasaje
m:integer)
Begin
x:=0;
x:=x+1;
m:=m+x + a(3);
x:=x*2;
a(3):=a(3) - 1;
m:=m+1;
End;
	Begin
	For i:=1 to 5 a(i):=1;
	x:=3;
	Uno(a(x));
	For i:=1 to 5 write (a(i));
	End.

a- Plantee diferencias, relacionada con la forma de implementación de cada uno y los resultados sobre este ejemplo considerando los siguientes tipos de pasajes parámetros: nombre, referencia y valor resultado.

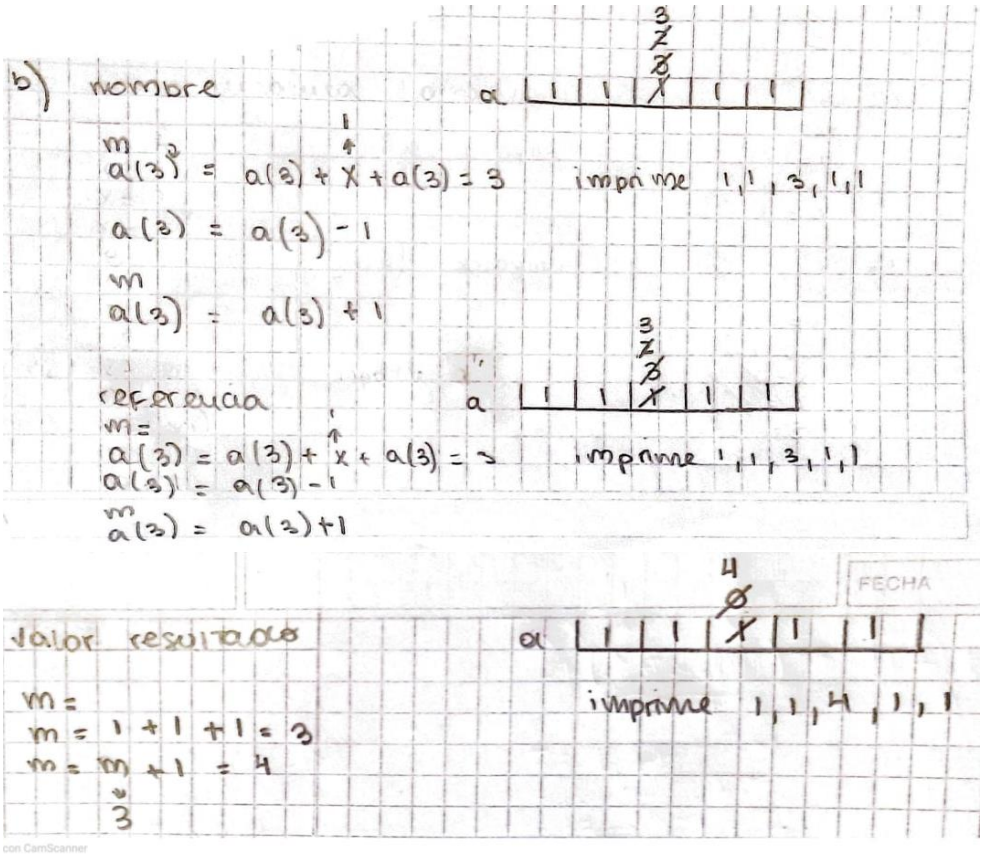
Nombre: El parámetro formal es sustituido textualmente por una expresión del parámetro real más un puntero al entorno del parámetro real. (se maneja una estructura aparte que resuelve esto). Se establece la ligadura entre parámetro formal y parámetro real en el momento de la invocación, pero la "ligadura de valor" se difiere hasta el momento en que se lo utiliza (la dirección se resuelve en ejecución). Distinto a por referencia. Es decir, no apunta a una dirección fija, puede ir cambiando (pero el nombre tiene que si ser el mismo)

Referencia: El parámetro formal será una variable local que contiene la dirección al parámetro real de la unidad llamadora que estará entonces en un ambiente no local. Cualquier cambio que se realice en el parámetro formal dentro del cuerpo del subprograma quedara registrado en el parámetro real.

Valor-Resultado: El parámetro formal es una variable local que recibe una copia (a la entrada) del contenido del parámetro real y el parámetro real (a la salida) recibe una copia de lo que tiene el parámetro formal. Básicamente lo que hace la rutina lo copio en la variable. Cada referencia al parámetro formal es una referencia local.



b- ¿Qué sucede si en Uno se agrega la siguiente declaración: x: integer? Indique el resultado para cada uno de los tipos de pasajes de parámetros (nombre, referencia y valor: resultado)



Ejercicio 10: Sea el siguiente un programa escrito en Pascal:

```
Program Uno;
        var x:integer;
        Function Dos:integer;
                begin
                x:= x + 1;
                return (x);
                end;
```

```
        Procedure Tres (pasaje x:integer);
                begin
                x:= x + 5;
                x:= Dos + 10;
                end;
begin
x:= 8; Tres(x);
write (x);
end.
```

a- Explique cómo simularía en Pascal el pasaje por valor-resultado y hágalo sobre este ejemplo.
Nota: No se pueden agregar más variables, ni cambiar el nombre de las que están.

```pascal
Program Uno;
var x:integer;

Function Dos:integer;
begin
  x:= x + 1;
  Dos := x;
end;

Function Tres (x: integer): integer;
```

```
begin
  x := x + 5;
  x := Dos + 10;
  Tres := x;
end;

begin
  x:= 8;
  x := Tres(x); // Llamada a la función Tres con pasaje por valor
  write (x);
end.
```

b- Transcriba este ejemplo en Ada de manera tal que el resultado de la ejecución sea diferente si el pasaje de parámetros es por referencia y luego por valor – resultado

```
1  with Ada.Text_IO;
2
3  procedure Uno is
4     x : Integer;
5
6     function Dos return Integer is
7     begin
8        x := x + 1;
9        return x;
10    end Dos;
11
12    procedure Tres (x : out Integer) is
13    begin
14       x := x + 5;
15       x := Dos + 10;
16    end Tres;
17
18    procedure Tres2 (x : in out Integer) is
19       temp : Integer;
20    begin
21       temp := x;
22       x := x + 5;
23       x := Dos + 10;
24       x := temp;
25    end Tres2;
26
27 begin
28    x := 8;
29    Tres(x);  -- Llamada a Tres con pasaje por referencia
30    Ada.Text_IO.Put("Resultado con pasaje por referencia: ");
31    Ada.Text_IO.Put(Integer'Image(x));
32
33    x := 8;
34    Tres2(x); -- Llamada a Tres2 con pasaje por valor-resultado
35    Ada.Text_IO.New_Line;
```

```
36    Ada.Text_IO.Put("Resultado con pasaje por valor-resultado: ");
37    Ada.Text_IO.Put(Integer'Image(x));
38 end Uno;
```

En Ada, el pasaje de parámetros se define mediante el uso de las palabras clave in, out y in out. La palabra clave in indica que el parámetro se pasa por valor, la palabra clave out indica que se pasa por referencia (es decir, se permite la modificación dentro del procedimiento o función), y la palabra clave in out indica que se pasa por valor-resultado.