Trabajo Práctico N° 06

Asignaciones

Recomendaciones Generales e Información

- Semántica de Asignación por copia **SAC**:
  - Se realiza una extracción de valor de la celda de memoria del lado derecho.
  - Ese valor extraído es copiado en la celda de memoria referenciada por el lado izquierdo.
  - Semejanza con **PCV**
- Semántica de Asignación por referencia **SAR**:
  - Se realiza una supresión de la extracción del valor del lado derecho y se obtiene una dirección.
  - Esa dirección es copiada en la celda de memoria referenciada por el lado izquierdo.
  - Semejanza con **PCR**
- Coerciones (Conversiones en **Algol**)

Desreferencing	Extracción de valor. Primero se extrae el valor de la variable del lado derecho y luego se hace la asignación **Algol →** implícitos y automáticos **C →** Hay un desreferencing implícito y para los demás se emplea *. & actúa como operador de supresión de extracción de valor
Desproceduring	Se convierte el procedimiento al valor que corresponde al llamado
Uniting	Solución de “Tipos Dinámicos” en lenguajes con Tipos Estáticos, pero no son dinámicos porque no cambian durante la ejecución. Válido sólo para las variables de tipo unión
Voiding	Convertir al tipo **void** en una asignación. Cuando no se precisa el valor se convierte en **void**
Widening	Convertir un tipo menos abarcativo en uno más abarcativo.
Rowing	Conversión de una variable de algún tipo en un arreglo unidimensional del mismo tipo

Ejercicio 1

Se tiene el siguiente fragmento de código escrito en lenguaje **XX**:

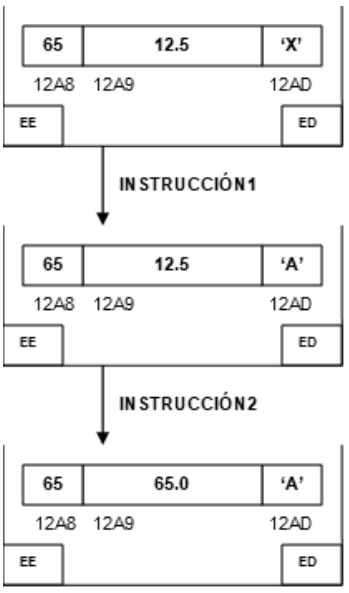
```
int main{
   int x,y,z;
   x := 10;
   y := 8;
   z := y;
   y := 10;
   z++;
   if (y>x) printf ("Y es mayor que X");
   if (y==x) printf ("X e Y son iguales");
   if (x>y) printf ("X es mayor que Y");
   return 0;
}
```

a) ¿Qué cartel se imprimirá por pantalla en el caso de **SAC**? Justifique construyendo la pila.
b) ¿Qué cartel se imprimirá por pantalla en el caso de **SAR**? Justifique construyendo la pila
c) ¿En qué sentencia/s se establece la diferencia del comportamiento entre los casos de los incisos a y b?
d) Reescriba el programa en el lenguaje C/C++ para simular ambos comportamientos.

Ejercicio 2 - Plan 2011

El siguiente código está escrito en el lenguaje **XX**, que *sólo acepta conversiones de tipos en forma explícita*:

```
proc YY(…)
begin
   short int a:=65;
   float b:=12.5;
   char c:=´X´;
   [INSTRUCCIÓN1]
   [INSTRUCCIÓN2]
end;
```



En tiempo de ejecución, se observan los cambios en los valores de las variables del registro de activación de **YY**:

a) Teniendo en cuenta que las instrucciones [**INSTRUCCIÓN1**] e [**INSTRUCCIÓN2**] son **asignaciones entre variables** ¿cómo sería el código de estas instrucciones?
b) ¿Qué diferencias existen entre la conversión presente en la [**INSTRUCCIÓN1**] y la conversión presente en la [**INSTRUCCIÓN2**]?
c) ¿La semántica es **SAC** o **SAR**? Justifique

Ejercicio 3

Dadas las siguientes declaraciones y asignaciones por copia en un lenguaje con semántica como la del lenguaje **C**, con conversiones implícitas y asignaciones múltiples:

```
int var1 = 7;
float var2 = var1;
float* var3 = &var2;
float** var4 = &var3;
float** var5 = var4;
```

i)

```
(var1, var2, var3, var4, var5) = (6,var1+1,&var1,&var3,var4)
```

ii)

```
var1 = 6;
var2 = var1+1;
var3 = &var1;
var4 = &var3;
var5 = var4;
```

a) Realizar un diagrama de las variables en memoria para cada caso i) y ii)

b) Muestre el diagrama modificado por las instrucciones y los valores de cada variable al final de la ejecución para i) y ii)

c) Indique todas las variables intervinientes en i) y ii)

Inciso i)											
Inciso ii)											

d) Exprese todas las declaraciones y asignaciones que sean posibles en **Algol** y realice un diagrama de las variables. Indique todas las coerciones para cada asignación

Ejercicio 4

Dados los siguientes conjuntos de declaraciones y asignaciones para un lenguaje tipo **Algol**:

```
double x = 3.5;
long int y = 4;
int z = 5;
```

a)	`x,y = y,x+y;`	b)	`x = y;` `y = x+y;`	c)	`x,z,y=y,x+y,z;`	d)	`x,x+y,y=y,z+y,z;`

i) ¿Qué diferencias sintácticas y semánticas existen entre los cuatro conjuntos de asignaciones?

ii) ¿Cuáles son los resultados para cada ítem?
¿Cómo influyen las diferencias semánticas sobre estos resultados?

iii) ¿En algún/algunos de los conjuntos el lenguaje utiliza variables temporales?
Indique cuáles variables y qué tipos/valores guardan.

iv) ¿Para algunos de los conjuntos de asignaciones se produce pérdida de precisión?
¿Dónde y por qué?

Nota: Asuma que el lenguaje admite asignaciones entre todos los tipos, no hay errores de conversión.

Ejercicio 5

Para cada uno de los siguientes programas:

J1 Java

```
class A {
   A n;
   A() {}
   A(A s) {
       n = s; //7
       }
};

class B {
   public A a1;
   B n;
   B() {
       a1 = new A(); //6
   }
};

public class parcial{
public static void main(String
[]args {
     A a2 = new A(); //1
     B b1 = new B(); //2
     b1.a1.n = new A(); //3
     b1.a1.n.n = b1.a1; //4
     b1.a1 = new A(a2); //5
   }
}
```

C1 C++

```
class A {
   public: A *n;
   A() {}
   A(A *s){
       n = s; //7
   }
};

class B {
   public: A *a1;
   B *n;
   B(){
       a1 = new A(); //6
   }
};


int main() {
   A *a2 = new A(); //1
   B *b1 = new B(); //2
   b1->a1->n = new A();
//3
   b1->a1->n->n = b1->a1;
//4
   b1->a1 = new A(a2); //5
}
```

C2 C++

```
class A {
   public: A *n;
   A() {}
   A(A *s) {
       n = s; //6
   }
};

class B {
  public: A a1;
   B *n;
   B() {}
};


int main() {
     A a2; //1
     B b1; //2
     b1.a1.n = &a2; //3
     b1.a1.n->n = &b1.a1;
//4
     b1.a1 = A(&a2); //5
}
```

a) Realizar un diagrama donde se muestre la ubicación de las variables en memoria y su evolución

b) Indicar, si existen, cuáles instrucciones generan garbage, justificando la respuesta.

c) Indicar para las asignaciones comentadas, si son **SAC** o **SAR**, justificando la respuesta.

d) Si el lenguaje tuviese garbage collection por marcado y borrado local, realizar un diagrama que muestre su ejecución y cuáles objetos marca como garbage

e) En el programa **C2**, indicar si es posible cambiar la declaración “A *n” por “A n”.
En caso afirmativo, realizar un diagrama del objeto **A** en memoria.
En caso negativo, justificar por qué.

Ejercicio 6

Considerando el siguiente fragmento de programa en un lenguaje tipo Algol (simil a C++).

a) Indicar (de la opción 1 a la 11) cuáles instrucciones son imposibles de compilar. Justifique su respuesta.
b) Indicar (de la opción 1 a la 11) cuáles instrucciones pueden provocar resultados inesperados o errores en tiempo de ejecución**.** Justifique su respuesta.
c) Indicar (de la opción 1 a la 11) cuáles instrucciones generan garbage.

```
fun1() {
```

```
    int a1 := 1;                    //1
    int *a2 := new (int);           //2
    *a2 := 2;                       //3
    static int a3;                  //4
    a3 := 3;                        //5
    static int *a4;                 //6
    *a4 := 4;                       //7
    a4 := a2;                       //8
    a1 := a3;                       //9
    *a2 := a3;                      //10
    a2 := &a1;                      //11
    imprime &a1;
    imprime &a2;
    imprime &a3;
    imprime &a4;
    imprime a2;
    imprime a4;
}
```

d) Suponiendo que se comentan las instrucciones que provocan errores y resultados inesperados, indicar cuál/es de las siguientes salidas pueden ser válidas.

S1	**S2**	**S3**
$100 $104 $-50000 $-50008 $100 $1000	$100 $-50000 $104 $-50008 $1000 $100	$100 $104 $1000 $-50000 $-50008 $100

e) Suponiendo que se comentan las instrucciones que provocan errores, asociar los siguientes códigos con sus correspondientes representaciones en memoria

M1 **Tupla: Variable, Dirección, Valores sucesivos (separados por !)**	**M2** **Tupla: Variable, Dirección, Valores sucesivos (separados por !)**	**M3** **Tupla: Variable, Dirección, Valores sucesivos (separados por !)**
a1, $100, 1 ! 3 a2, $104, $1000 ! $100 a3, $50000, 3 a4, $50008, null o basura ! $1000	a1, $1000, 1 ! 3 a2, $104, $1000 ! $1000 a3, $1004, 1 ! 3 a4, $50008, null o basura ! $1000	a1, $100, 1 ! 3 a2, $1000, $1000 ! $100 a3, $50000, 1 ! 3 a4, $104, null o basura ! $1000

f) Indicar cuántos desreferencing del lado derecho se realizan en cada una de las instrucciones comentadas.

g) Responder verdadero o falso cuando corresponda, o elegir la opción correcta justificando muy brevemente sus respuestas.

1	Las variables a3 y a4 son las de mayor tiempo de vida	
2	La variable declarada a2 es: -dinámica anónima -semidinámica -semiestática	
3	Al finalizar el programa, la variable a4 contiene: : - la dirección de una variable estática - la dirección de una variable semidinámica - la dirección de una variable dinámica anónima	
4	Al finalizar el programa el contenido de la variable a2 es la dirección	

	de una variable dinámica con nombre	
5	Si se reemplaza la asignación //11 por “a2 = &a3;” daría error de compilación porque: - una variable estática no posee dirección - son tipos incompatibles - el operador & no aplica a una variable estática - ninguna de las anteriores	
6	Existen en memoria dos celdas con variables dinámicas anónimas	

Ejercicio 7

Se tiene el siguiente programa en un lenguaje tipo **Algol** con *conversiones implícitas y asignaciones múltiples*. La semántica de las asignaciones simples es la misma que la de **C++**.

```
int main() {
 int x = 0;              //[1]
 int y = --x;            //[2]
 int z = y++;            //[3]
 float j = x+1.5;        //[4]
 int * a = &z;           //[5]
 int ** b = &a;          //[6]
 float p = (**b)--;      //[7]
 int w = j = --y;        //[8]
 print(x,y,z,j,p,w);     //[9] imprime todas las variables pasadas por parámetro
 x,y,z = y+x,x,y;        //[10]
 print(x,y,z,**b);       //[11] imprime todas las variables pasadas por parámetro
}
```

i) Realizar un diagrama de las variables en memoria mostrando los cambios de las mismas a lo largo del programa.
ii) Indicar el resultado de todas las impresiones.
iii) Indicar dónde y para qué se utilizan variables auxiliares.
iv) Indique, para cada línea, las coerciones que se realizan, especificando sobre qué variable de la asignación se hace la conversión.
v) Explique qué parte de la asignación //4 se resuelve en compilación y qué parte en ejecución.
vi) ¿Qué cambiaría en la forma de resolver las asignaciones si el lenguajes fuese dinámico?
vii) Responda si las siguientes afirmaciones son Verdaderas o Falsas. Justifique todas sus respuestas
   a) Exceptuando la línea `[1]`, en todas las asignaciones hay al menos un desreferencing.
   b) Los resultados del print de la línea `[9]` serían los mismos si las asignaciones simples entre variables fueran por referencia.
   c) Si la línea `[10]` se reemplazara por

```
x = y+x;
y = x;
z = y;
```

   habría diferencias sintácticas pero no semánticas.

   d) Si no se quiere tener garbage al finalizar el programa, se debe liberar la memoria apuntada por los punteros **a** y **b** antes de terminar el bloque **main**.

Ejercicio 8 - Plan 2011

Sea un lenguaje en el que es posible definir en forma separada las variables reales de aritmética **BCD** y de aritmética **IEEE**.

i) Sean las siguientes declaraciones:

```
realBCD a,b,c,d;
realIEEE x,y;
integer m,n;
```

En los siguientes fragmentos de programa, los resultados esperados son siempre `m=33` y `n=35`.
Indique si esto se logra.
Si no dan estos resultados explique por qué.

Caso a)	**Caso b)**

`a:= 4.125;`
`b:= 8.0;`
`c:= 0.175;`
`d:= 200;`
`x:=a;`
`y:=c;`
`m:= x * b;`
`n:= y *d;`
	`a:= 4.125;`
	`b:= 8.0;`
	`c:= 0.175;`
	`d:= 200;`
	`x:=a;`
	`y:=b;`
	`m:= x * y;`
	`n:= c *d`

ii) En este lenguaje se tiene

	Posibles Resultados
`a:=2;` `b:=7;`	`a/b → 0,28571428` `a/b → 0,28571400`
(en ambos casos sólo visualizando 8 dígitos)	

Considerando que el resultado es el número real `0,285714` ... (período `285714`)

a. ¿Con qué aritmética fueron declaradas las variables para cada resultado? Justifique.
b. ¿Cuál es más precisa y por qué?
c. ¿Cuáles lenguajes podrían dar `0,28571428` y cuáles darían `0,28571400`?

Ejercicio 9

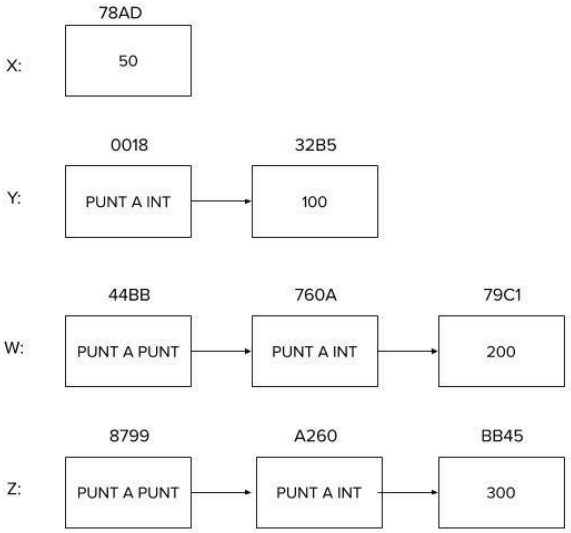
Se tienen cuatro variables **A**, **B**, **C** y **D** en **Algol**.
Realizar las declaraciones de cada variable de tal manera que se cumplan, al mismo tiempo, las siguientes restricciones en las asignaciones:

a) Casting, widening y un desreferencing entre **A** y **B**
b) Tres desreferencing entre **C** y **A**
c) Desproceduring entre **D** y **B**
d) Casting, Desproceduring y desreferencing entre **C** y **D**

Para cada inciso indicar cómo y en cuál variable se producen las conversiones implícitas requeridas y detallar los casos en que se necesitan variables auxiliares para la asignación.

Ejercicio 10

Dadas las siguientes variables en las cuales las celdas finales de los datos contiene los valores 50, 100, 200 y 300.



a) Indique las declaraciones de las mismas en **Algol** y **C**.
b) Con las variables anteriores, complete el siguiente cuadro:

HACER QUE:	ASIGNACIÓN (si no es posible diga por qué)		CANTIDAD DESREFERENCING	GRÁFICO RESULTANTE (dibujar solo lo que cambia)
	ALGOL	C		
El puntero referenciado por Y apunte al entero de valor 300				
El entero referenciado por X adquiera el entero referenciado por W				
La dirección 8799 contenga el valor 50				
El entero apuntado por W contenga el contenido de X incrementado en 1				
El puntero al entero referenciado por Z contenga 79C1				

Ejercicio 11

Si las variables **u**, **v** y **w** se encuentran en las direcciones de memoria:

w	**CDA11**$_h$
v	**12340**$_h$
u	**AB0F2**$_h$

Y las definiciones fueron:

En **ALGOL**	En **C**

`int w;`
`ref int v;`
`ref ref int u;`
	`int w;`
	`int *v;`
	`int **u;`

Y si a su vez el contenido de cada una de las variables es:

w	**00017**
v	**44BBB**
u	**8788A**

a) Escribir las definiciones que permita que **v** contenga **CDA11** en **ALGOL** y **C**.

b) Es posible que **u** contenga el valor **12340**. Explicar.
c) Es posible que **u** contenga el valor **CDA11**. Explicar.
d) Escribir las definiciones que permitan hacer que **u** acceda al número **00017**.
e) Utilizando **u** y el resultado de **d** incrementar **17** en **1**.

Aclaración: Los valores no deben ser asignados sino tomados desde las variables ya existentes

Ejercicio 12

En un lenguaje con binding estático de tipos, se tiene el siguiente programa con las siguientes declaraciones

```
Programa Mi programa
Comienzan las declaraciones
   Se declaran las variables i1, i2 de tipo entero;
   Se declaran las variables f1, f2 de tipo flotante;
   Se declaran las variables pi1, pi2 de tipo puntero a entero;
   Se declaran las variables ppi1, ppi2 de tipo puntero a puntero a entero;
   Se declaran las variables pppi1, pppi2 de tipo puntero a puntero a puntero a entero;
   Se declaran las variables pf1, pf2 de tipo puntero a flotante;
   Se declaran las variables ai1 de tipo arreglo de 10 entero;
   Se declaran las variables af1 de tipo arreglo de 10 flotante;
Finalizan las declaraciones;
```

y las siguientes instrucciones ejecutables.

a) Escribir cada instrucción como su equivalente en C++ en la siguiente lista.

#	Equivalente a C++	Instrucción
1		i1 se vuelve 1;
2		i2 se vuelve i1;
3		f1 se vuelve i1;
4		f2 se vuelve f1;
5		pi1 se vuelve la dirección de i1;
6		pi2 se vuelve la dirección de un bloque nuevo de tipo entero;
7		ppi1 se vuelve la dirección de un bloque nuevo de tipo puntero a entero;
8		el contenido de lo apuntado por ppi1 se vuelve la dirección de un bloque nuevo de tipo entero;
9		ppi2 se vuelve la dirección de pi1;
10		El contenido de lo apuntado por el contenido de lo apuntado por ppi2 se vuelve i1;
11		El contenido de lo apuntado por el contenido de lo apuntado por ppi2 se vuelve f2;
12		pppi1 se vuelve la direccion de un bloque nuevo de tipo puntero a puntero a entero;
13		pppi2 se vuelve pppi1;
14		pf1 se vuelve la dirección de un bloque nuevo de tipo flotante;
15		el contenido de lo apuntado por pf1 se vuelve 1;
16		pf2 se vuelve la dirección de f2;
17		ai1 se vuelve 1;
18		af1 se vuelve ai1;

b) Indicar para cada instrucción, cuántos dereferencing implícitos y explícitos existen en el lado derecho, si existe widening y si existe rowing, asumiendo para las asignaciones, un comportamiento como el de la mayoría de los lenguajes (C, C++, etc).

#	DEREFERENCING IMPLICITO	DEREFERENCING EXPLICITO	WIDENING	ROWING
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
…				
17				
18				

c) Suponiendo que el lenguaje posee dereferencing implícitos automáticos (como Algol), indicar cuántos dereferencing hay para las siguientes instrucciones:

INSTRUCCIÓN	DEREFERENCING
i1 se vuelve 1	
i1 se vuelve pi1	
pi1 se vuelve pi2	
pi1 se vuelve ppi1	
pi1 se vuelve pppi1	

Ejercicio 13

Considere el siguiente código de un programa completo sin errores de sintaxis que al ejecutarlo en cinco lenguajes diferentes se obtienen las salidas indicadas:

Código	
1	a = 2
2	b = “2”
3	c = 3.1
4	print(a)
5	print(a+c)
6	print(a+b)
7	b = “asd”
8	c = a
9	print(c.type)

Salidas	
S1	2 5.1 22 float
S2	error linea 5
S3	2 error linea 6
S4	2 5 4 int
S5	error linea 7

a) Responda las siguientes preguntas, **justificando brevemente** en cada caso

i. ¿Cuál/es de la/s siguientes características tienen en común los cinco lenguajes?

1. Lenguaje con tipos estáticos
2. Lenguaje interpretado
3. Compatibilidad por nombre
4. Inferencia de tipos

ii. ¿Cuál/es característica/s del lenguaje se evidencia/n en la salida S5?

1. Incompatibilidad de tipos
2. Lenguaje con tipos dinámicos
3. Variables pseudo-constantes o de solo lectura
4. Alcance dinámico.

Código	
1	`a = 2`
2	`b = "2"`
3	`c = 3.1`
4	`print(a)`
5	`print(a+c)`
6	`print(a+b)`
7	`b = "asd"`
8	`c = a`
9	`print(c.type)`

Salidas	
S1	`2 5.1 22 float`
S2	`error linea 5`
S3	`2 error linea 5`
S4	`2 5 4 int`
S5	`error linea 7`

iii. Si al cambiar la línea 5 por:

print(a + int(c))

S2 cambia por

2 5 error línea 6

¿Cuáles de las siguientes afirmaciones sobre el lenguaje son verdaderas?

1. El lenguaje tiene tipos dinámicos
2. El lenguaje posee el mismo tipo para las variables enteras y de punto flotante
3. El lenguaje tiene conversiones explícitas
4. El lenguaje es interpretado

iv. Para cada salida ¿en qué líneas cambian los tipos de a, b o c?

v. Para cada salida ¿en cuáles instrucciones se utilizan variables auxiliares? ¿Para qué?

vi. ¿Cuáles coerciones se producen en cada línea para la salida S1?

vii. Si el lenguaje en el que se escribió el código dado tuviera una semántica de compatibilidades de tipos idéntica a ADA y las variables **a**, **b** y **c** fueran de los siguientes tipos:

```
a : Integer;
b : String(1 .. 1);
c : Float;
```

¿Qué errores de compilación y/o ejecución se producirían y por qué? Indique las líneas en las que suceden.

b) Complete la siguiente tabla con **SÍ / NO / NS (No se puede saber)**.

Justifique brevemente cada caso.

Salida	**¿Posee tipos dinámicos?**	**¿Posee conversiones implícitas?** ***(Indicar los tipos implicados en la conversión)***	**¿Posee diferentes semánticas de operador + para cada tipo? (*Indicar los tipos implicados en la suma en esos casos*)**

S1			
S2			
S3			
S4			
S5			

Ejercicio 14

Dado el siguiente código en un lenguaje que soporta retorno de múltiples valores de funciones y asignaciones múltiples:

```
func f(x int, y int, z int) (int, int, int) {
        x, y, z = x+1, x+y, x+y+z
        return x, y, z
}

func main() {
        var a, b, c = 0, 0, 0
        a, b, c = f(a, b, c)
        fmt.Println(a, b, c)
}
```

a) Indicar qué imprime el programa
b) Indicar qué imprime el programa si el lenguaje no crea variables temporales para la asignación
c) Indicar qué imprime el programa si el lenguaje soporta asignaciones múltiples, no soporta retorno de valores múltiples y tiene pasaje por Copia Valor, Copia Valor Resultado, Copia resultado, Referencia y Nombre

Indicar en cada caso cuáles variables temporales se crean.
Fundamentar las respuestas.

Ejercicio 15

Dados los siguientes programas en Python y C# (Es posible ejecutarlos en compiladores/intérpretes locales o en los enlaces indicados a herramientas online)

[Python](#)

```
class Entero:
    valor = 3
a = Entero()
b = Entero()
b = a
a.valor = 4
print (a.valor)
print (b.valor)
x = 3
y = x
x = 4
print (y)
```

[C#1](#)

```
using System;

public class Program
{
class EnteroClase {
   public int valor=3;
}
public static void Main(string[] args)
{
  EnteroClase a = new EnteroClase();
  EnteroClase b = new
EnteroClase();
  b=a;
  a.valor = 4;
  Console.Write(b.valor);
  int x = 3;
  int y = x;
```

[C#2](#)

```
using System;

public class Program
{
struct EnteroStruct {
  public int valor;
 }
public static void Main(string[] args)
{
  EnteroStruct c;
  c.valor = 3;
  EnteroStruct d = c;
  c.valor = 4;
  Console.Write(d.valor);
  EnteroStruct e = new    EnteroStruct();
  e.valor = 3;
  EnteroStruct f = e;
```

	x = 4;
	Console.Write(y);
	Console.Write("\n");
	}
	}
		e.valor = 4;
		Console.Write(f.valor);
		}
		}

a) Indicar qué imprime cada programa
b) Realizar un diagrama de memoria da cada programa
c) Para cada asignación entre variables, indicar si son por copia o por referencia

Ejercicio 16 - Plan 2011

Responda si las siguientes afirmaciones son **verdaderas** o **falsas** justificando las respuestas:

a) Un lenguaje con representación **BCD** posee más precisión que un lenguaje con representación **IEEE** en el almacenamiento de valores de punto flotante.
b) Un lenguaje con representación **IEEE** no es capaz de almacenar el valor **0.1** pero sí múltiplos de **10** del mismo valor.
c) La causa de los problemas, debido a la representación, es que el valor a almacenar es periódico tanto en el sistema decimal como en binario.
d) Los problemas, debidos al sistema de representación, utilizado para punto flotante se extienden también al almacenamiento de valores enteros.
e) En un lenguaje tipo **Algol** sería posible cambiar el tipo de representación de una variable en cualquier instrucción.
f) En un lenguaje con tipos dinámicos sería posible cambiar el tipo de representación de una variable en cualquier instrucción