COLECCIÓN DE EXÁMENES DE LA ASIGNATURA APLICACIÓN DE ORDENADORES 2005 – 2011

Fundamentos de programación y algoritmia

EMMANUEL DAVID QUEREDA GÓMEZ
LUIS BAÑÓN BLÁZQUEZ

ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE ALICANTE

Título: Colección de exámenes de la asignatura Aplicación de Ordenadores 2005 – 2011. Fundamentos de programación y algoritmia.

Autores: Emmanuel David Quereda Gómez y Luis Bañón Blázquez

WEB de los autores: https://sites.google.com/site/davidquereda/ y http://personal.ua.es/lbanon/

Dpto. de Ing. de la Construcción, Obras Públicas e Infraestructura Urbana
Escuela Politécnica Superior - Universidad de Alicante
Edificio Politécnica II
Ctra. San Vicente del Raspeig, s/n
03690 - San Vicente del Raspeig - ESPAÑA

PREÁMBULO

Esta divulgación docente recoge los ejercicios resueltos de aquellos exámenes realizados durante los curso 2005-2011, correspondientes a la asignatura *Aplicación de Ordenadores* de 3º curso de la titulación de Ingeniería Técnica de Obras Públicas de la Escuela Politécnica Superior de la Universidad de Alicante. El contenido de esta colección de exámenes versa sobre fundamentos de programación y algoritmos.

Antes de comenzar a emplear esta colección de ejercicios, es conveniente recordar que la solución para cada ejercicio casi nunca es única y, seguramente, la propuesta en este texto pudiera no ser la “óptima”, aunque sí correcta y razonable.

Espero que esta publicación sea de provecho como material de apoyo para preparar la asignatura a aquellos que la estáis cursando o para aquellos que deseáis adquirir práctica en esta materia. Aunque se ha puesto el máximo cuidado, es posible que se detecten erratas, agradeceré a cualquiera que advierta alguna errata en las soluciones propuestas y que me la comunique para subsanarla.

Alicante, 10 de octubre de 2011

Emmanuel David Quereda Gómez
Profesor coordinador de la asignatura

ÍNDICE

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JULIO 2.011 (Examen tipo A) (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	
D.N.I.:	

P1. Ejercicio teórico.

a) Dadas las funciones siguientes:

Función ***F1*** (x : real): entero
Comienzo
F1 ← x + 0.5
Fin-función.

Función ***F2*** (x : entero): real
Comienzo
F2 ← x * 3.0
Fin-función.

Por medio de una traza indicar el valor de R al ejecutar la siguiente línea de un algoritmo: **(1 punto)**
R ← **F2** (**F1** (**F1** (3.5))) 'Siendo R de tipo real

X	F1
3,5	4
4,0	4

X	F2
4	12,0

R
12,0

Comprobación realizada con Visual Basic:
En un formulario introducir una caja de texto (TextBox) de nombre txtResultado. Introducir un botón (CommandButton) de nombre cmdCalcular y copiar el código siguiente en la ventana 'Ver código' de Visual Basic.

```
Option Explicit
Dim R As Double

Private Sub cmdCalcular_Click()
R = F2(F1(F1(3.5)))
txtResultado.Text = R
End Sub

Function F1(x As Double) As Integer
   F1 = x + 0.5
End Function

Function F2(x As Integer) As Double
   F2 = x * 3
End Function
```

b) Deducir el resultado de las expresiones paso a paso como operaría un ordenador. **(1 punto)**

1. – 4 * 7 + 2 ^3 / 4 – 5

- 4 * 7 + 2 ^3 / 4 -5
- 4 * 7 + 8 /4 –5
- 28 + 8 / 4 - 5
- 28 + 2 - 5
- 31

2. 7 * 10 –15 mod 3 *4 + 9

7 * 10 –15 mod 3 * 4 + 9
70 –15 mod 3 * 4 + 9
70 – 0 * 4 + 9
70 - 0 + 9

79

3. no (z > 14) para z = 7

no(7 >14)
no(falso)verdadero

4. (4,5 > x) y (z < x + 7,5) para x = 7 y z = 5

(4,5 > 7) y (5 < 7 + 7,5)
(falso) y (5 < 7 + 7,5)
(falso) y (5 < 14,5)
(falso) y (verdadero)
falso

c) Obtener el número binario de la expresión $D_2 = 22_{16} / 6_{16}$ (**2 punto**)
Una forma de resolver este problema sería:
$22_{16 \rightarrow 2}$ si aplicamos el teorema fundamental de la numeración $2 \times 16^0 + 2 \times 16^1 = 2 + 32 = 34_{10}$

$6_{16 \rightarrow 2}$ si aplicamos el teorema fundamental de la numeración $6 \times 16^0 = 6_{10}$

34/6 da de cociente entero 5 y de resto entero 4

	Cociente de la división entera	Resto de la división entera
5/2	2	1
2/2	1	0
1/2	0	1

si recogemos el último cociente y los restos desde abajo hacía arriba tenemos como cociente entero de la división $0101_2 = 5_{16}$

	Cociente de la división entera	Resto de la división entera
4/2	2	0
2/2	1	0
1/2	0	1

si recogemos el último cociente y los restos desde abajo hacia arriba tenemos como resto entero de la división $0100_2 = 4_{16}$

La división se realiza de forma semejante al decimal, con la salvedad que las multiplicaciones y restas internas del proceso de la división se realizan en binario.

```
100010 | 110
- 110    101
  1010
 - 110
   100
```

P2. Realizar la traza del siguiente algoritmo, indicando los valores que toman las variables i, j, a(i), a(j), a(j +1), a(4) y suma. **(1 punto)**

```
Algoritmo Traza
variables
        entero: a(i), a(j), a(j +1); a(4); i , j; suma; b      // No se ha indicado la descripción por no ser necesaria
comienzo
        hacer suma ← 1 + suma - b
        para i ← 1 hasta 3
                para j ← 1 hasta 2
                        hacer   a(i) ← i + i
                                a(j) ← j + j
                                a(j +1) ← i + j
                                a(4) ← 0
                                suma ← suma + a(i + 1) + a(j) + b
                siguiente j
        siguiente i
fin
```

suma	i	j	a(i)	a(j)	a(j +1)	a(4)	suma
1	1 → 3	1 → 2	a(1) = 2	a(1) = 2	a(2) = 2	0	5
		2	a(1) = 2	a(2) = 4	a(3) = 3	0	13
	2	1 → 2	a(2) = 4	a(1) = 2	a(2) = 3	0	18
		2	a(2) = 4	a(2) = 4	a(3) = 4	0	26
	3	1 → 2	a(3) = 6	a(1) = 2	a(2) = 4	0	28
		2	a(3) = 6	a(2) = 4	a(3) = 5	0	32

a) ¿Cuánto vale la variable b al finalizar el algoritmo?, explica el por qué de este valor. (**0,5 puntos**)

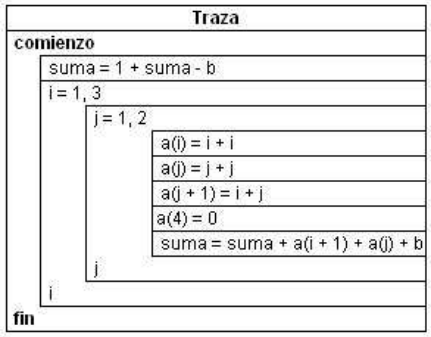
Una vez que se declara una variable, el sistema le asigna un valor por defecto asociado al tipo de variable, ya que la variable está creada y además con su dato por defecto. Así, que si se usa la variable 'b', ésta contiene en memoria un dato, que está claro que si uno no lo inicializa al valor que le interesa contendrá el de defecto, en este caso b = 0 por ser de tipo entero la variable 'b'. Como ha la variable b no se le asigna ningún valor durante el desarrollo del algoritmo, el valor final al finalizar el algoritmo coincide con el de la declaración de la variable, b = 0.

Nota: No es normal usar el valor por defecto, ya que casi siempre se necesita si la variable es de tipo entero que sea distinto a cero o en algunos lenguajes es obligatorio inicializar la variable.

Código en Visual Basic del algoritmo anterior:
Con un formulario sin objetos, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Private Sub Form_Load()
    Dim a(4) As Integer
    Dim i As Integer
    Dim j As Integer
    Dim suma As Integer
    Dim b As Integer

    suma = 1 + suma + b
    For i = 1 To 3
        For j = 1 To 2
        a(i) = i + i
        MsgBox ("a(" + Str(i) + ") = " + Str(a(i)))         'Muestra el valor de la traza
        a(j) = j + j
        MsgBox ("a(" + Str(j) + ") = " + Str(a(j)))         'Muestra el valor de la traza
        a(j + 1) = i + j
        MsgBox ("a(" + Str(j + 1) + ") = " + Str(a(j + 1)))          'Muestra el valor de la traza
        a(4) = 0
        MsgBox ("a(4) = " + Str(a(4)))          'Muestra el valor de la traza
        suma = suma + a(i + 1) + a(j) + b
        MsgBox ("Suma = " + Str(suma))          'Muestra el valor de la traza
        Next j
    Next i
End Sub
```

b) Transformar el algoritmo del apartado anterior en diagrama de N - S. **(1 punto)**



P3. Diseñar un algoritmo en pseudocódigo que permita aceptar los (n) elementos de un vector (V) de componentes enteros y positivos y que permita localizar al tiempo que se introducen los elementos del vector, los números primos que figuran en el, mostrando su valor y la posición que ocupan cada uno en el vector. Como única estructura de repetición solamente se podrá usar la de condición inicial. Deberán disponerse todos los filtros necesarios para que no se produzcan errores. Realizar el algoritmo usando por lo menos las variables dadas entre paréntesis del enunciado. **(2,5 puntos)**

Número primo es aquel que solo es divisible por si mismo y por la unidad. Por convenio el 1 no se considera primo.
Ejemplo:
V(0, 3, 11, 4, 1) con números primos: 3 y 11, ocupando las posiciones respectivamente 2 y 3

```
Algoritmo Elementos primos de un vector
variables
        entero: i (contador de elementos); n (longitud del vector); V(i) (vector con números enteros positivos); c (contador
        de números)
comienzo
        mientras n <= 0
            mostrar "Introduzca el número de elementos del vector V:"
            aceptar n
                si n <=0 entonces
                    mostrar "El numero de elementos del vector debe ser superior a cero"
                fin-si
        fin-mientras
        mientras i < n
            hacer i ← i + 1
            mostrar "Introduzca el valor del elemento " + i + ":"
            aceptar V(i)
            si V(i) < 0 entonces
                mostrar "El número " + V(i) + " es negativo, se convierte a positivo."
                hacer V(i)  ← abs(V(i))
            fin-si
            hacer c ← 2          // Hemos quedado que por convenio el 1 no es primo
            mientras c < V(i) y V(i) mod (c) <> 0   // Se comprueba si es divisor de V(i)
                hacer c ← c + 1
            fin-mientras
            si c = V(i) entonces
                mostrar "El elemento que ocupa la posición " + i + "de valor " + V(i) + "es primo"
            si-no
                mostrar "El elemento que ocupa la posición " + i + "de valor " + V(i) + "no es primo"
            fin-si
        fin-mientras
fin
```

a) Realizar la traza para n = 6 con V = $(0,3,11,4,1,2)$. **(1 punto)**

n	i	V(i)	c	c	solución
6	1	0	2		Elemento 1 de valor 0 no es primo
	2	3	2	3	Elemento 2 de valor 3 es primo
	3	11	2	3	
				4	
				5	
				6	
				7	
				8	
				9	
				10	
				11	Elemento 3 de valor 11 es primo
	4	4	2		Elemento 4 de valor 4 no es primo
	5	1	2		Elemento 5 de valor 1 no es primo
	6	2	2		Elemento 6 de valor 2 es primo

Código en Visual Basic del algoritmo anterior:
Con un formulario sin objetos, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Option Explicit

Dim i As Integer  'contador de elementos
Dim n As Integer  'longitud del vector
Dim c As Integer  'contador de números
Dim V() As Integer      'vector con números enteros positivos

Private Sub Form_Load()
   While n <= 0
      n = InputBox("Introduzca el número de elementos del vector V:")
      If n <= 0 Then
         MsgBox ("El número de elementos del vector debe ser superior a cero")
      End If
   Wend
   ReDim V(n)
   While i < n
      i = i + 1
      V(i) = InputBox("Introduzca el valor del elemento " + Str(i) + " :")
      c = 2
      If V(i) < 0 Then
```

```
            MsgBox ("El número " + Str(V(i)) + " es negativo, se convierte a positivo.")
            V(i) = Abs(V(i))
        End If
        While (c < V(i)) And (V(i) Mod (c) <> 0)
            c = c + 1
        Wend
        If c = V(i) Then
            MsgBox ("El elemento que ocupa la posición " + Str(i) + " de valor " + Str(V(i)) + " es primo")
        Else
            MsgBox ("El elemento que ocupa la posición " + Str(i) + " de valor " + Str(V(i)) + " no es primo")
        End If
    Wend
End Sub
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JUNIO 2.011 (Examen tipo A) (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	
D.N.I.:	

P1. Ejercicio teórico.

a) ¿Qué es un interruptor en un algoritmo? **(1 punto)**

Un interruptor o también llamado switch, es una variable de control que puede tomar los valores verdad y falso o 0 y 1 (si se declara de tipo lógico) a lo largo de la ejecución de un algoritmo o programa, comunicando así información de una parte a otra del mismo. Pueden ser utilizados para el control de bucles de repetición.

b) Escribe un ejemplo de algoritmo en pseudocódigo que use un interruptor para permitir salir de un bucle de repetición de forma controlada. **(1 punto)**

```
Algoritmo Uso de interruptor
variables
        entero: n, c, V(n)      // Para el ejemplo no es necesario describir las variables
        lógica: sw  (Interruptor)   // Para el ejemplo no es necesario describir la variable
comienzo
        aceptar n
        hacer sw ← "verdad"
        mientras c <= n y sw = "verdad"
            hacer c ← c + 1
            aceptar V(c)
            si V(c) < 0 entonces
                hacer sw ← "falso"  // Testigo para no seguir comprobando
            fin-si
        fin-mientras
        si sw = "verdad"  entonces
            mostrar "Entrada de datos correctos"
        si-no
            mostrar "Entrada de datos interrumpida por valor negativo"
        fin-si
fin
```

c) Obtener el número binario de la expresión $R_2 = 99D - 572_{16}$ **(2 punto)**

Para resolver el problema hay varias formas de hacerlo.
La forma más rápida de hacerlo pero que exige conocer la tabla de conversión entre hexadecimal y binario sería:

$$\begin{array}{rrr} 9 & 9 & D_{16} \\ -\ 5 & 7 & 2_{16} \\ \hline 4 & 2 & B_{16} \end{array}$$

es decir $(9 - 5)_{16} = 4_{16}$
$(9 - 7)_{16} = 2_{16}$
$(13 - 2)_{10} = 11_{10} = B_{16}$

como $4_{16} = 0100$
$2_{16} = 0010$
$B_{16} = 1011$ **obtenemos** 010000101011_2

Otra forma más rigurosa y más lenta de resolver el problema sería:
$99D_{16 \to 2}$ sustituimos su equivalente D por el valor que ocupa como dígito, 13 y aplicamos el teorema fundamental de la numeración $13 \times 16^0 + 9 \times 16^1 + 9 \times 16^2 = 13 + 144 + 9 \times 256 = 13 + 144 + 2304 = 2461_{10}$

	Cociente de la división entera	Resto de la división entera
2461/2	1230	1
1230/2	615	0
615/2	307	1
307/2	153	1
153/2	76	1
76/2	38	0
38/2	19	0
19/2	9	1
9/2	4	1
4/2	2	0
2/2	1	0
1/2	0	1

Por tanto, si recogemos el último cociente y los restos desde abajo hacia arriba tenemos $0100110011101_2 = 99D_{16}$

$572_{16 \rightarrow 2}$ si aplicamos el teorema fundamental de la numeración $2 \times 16^0 + 7 \times 16^1 + 5 \times 16^2 = 2 + 112 + 5 \times 256 = 2 + 112 + 1280 = 1394_{10}$

	Cociente de la división entera	Resto de la división entera
1394/2	697	0
697/2	348	1
348/2	174	0
174/2	87	0
87/2	43	1
43/2	21	1
21/2	10	1
10/2	5	0
5/2	2	1
2/2	1	0
1/2	0	1

si recogemos el último cociente y los restos desde abajo hacia arriba tenemos $010101110010_2 = 99D_{16}$

Los ceros a la izquierda no tienen valor.

La resta 0 - 1 se resuelve, igual que en el sistema decimal, tomando una unidad prestada de la posición siguiente: 10 - 1 = 1 y me llevo 1, lo que equivale a decir en decimal, 2 - 1 = 1. Esa unidad prestada debe devolverse, sumándola, a la posición siguiente.
A pesar de lo sencillo que es el procedimiento, es fácil confundirse. Tenemos interiorizado el sistema decimal y hemos aprendido a restar mecánicamente, sin detenernos a pensar en el significado del arrastre. Para simplificar las restas y reducir la posibilidad de cometer errores podemos dividir los números largos en grupos.

$$\begin{array}{r} 100110011101 \\ -\ 010101110010 \\ \hline 010000101011 \end{array} = \begin{array}{rrr} 1001 & 1001 & 1101 \\ -\ 0101 & -\ 0111 & -\ 0010 \\ \hline 0100 & 0010 & 1011 \end{array}$$

Por tanto, $R_2 = 99D_{16} - 572_{16} = 010000101011_2$

P2. Realizar la traza del siguiente algoritmo, indicando los valores que toman las variables i, j, a(i), a(j), a(j +1) y suma. **(1 punto)**

```
Algoritmo Traza
variables
        entero: a(i), a(j), a(j +1); i , j; suma; b      // No se ha indicado la descripción por no ser necesaria
comienzo
        hacer suma ← 1 + suma - b
        para i ← 1 hasta 3
                para j ← 1 hasta 2
                        hacer   a(i) ← i - j
                                a(j) ← j - i
                                a(j +1) ← i + j
                                suma ← suma + a(i) + a(j) + b
                siguiente j
        siguiente i
fin
```

suma	i	j	a(i)	a(j)	a(j +1)	suma
1	1 → 3	1 → 2	a(1) = 0	a(1) = 0	a(2) = 2	1
		2	a(1) = -1	a(2) = 1	a(3) = 3	1
	2	1 → 2	a(2) = 1	a(1) = -1	a(2) = 3	3
		2	a(2) = 0	a(2) = 0	a(3) = 4	3
	3	1 → 2	a(3) = 2	a(1) = -2	a(2) = 4	3
		2	a(3) = 1	a(2) = -1	a(3) = 5	7

a) ¿Cuánto vale la variable b al finalizar el algoritmo?, explica el por qué de este valor. **(0,5 puntos)**

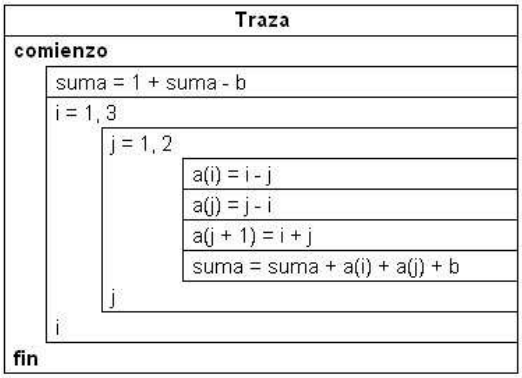
Una vez que se declara una variable, el sistema le asigna un valor por defecto asociado al tipo de variable, ya que la variable está creada y además con su dato por defecto. Así, que si se usa la variable 'b', ésta contiene en memoria un dato, que está claro que si uno no lo inicializa al valor que le interesa contendrá el de defecto, en este caso b = 0 por ser de tipo entero la variable 'b'. Como la variable b no se le asigna ningún valor durante el desarrollo del algoritmo, el valor final al finalizar el algoritmo coincide con el de la declaración de la variable, b = 0.

Nota: No es normal usar el valor por defecto, ya que casi siempre se necesita si la variable es de tipo entero que sea distinto a cero o en algunos lenguajes es obligatorio inicializar la variable.

Código en Visual Basic del algoritmo anterior:
Con un formulario sin objetos, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Private Sub Form_Load()
    Dim a(3) As Integer
    Dim i As Integer
    Dim j As Integer
    Dim suma As Integer
    Dim b As Integer
    suma = 1 + suma + b
    For i = 1 To 3
        For j = 1 To 2
        a(i) = i - j
        MsgBox ("a(" + Str(i) + ") = " + Str(a(i)))       'Muestra el valor de la traza
        a(j) = j - i
        MsgBox ("a(" + Str(j) + ") = " + Str(a(j)))       'Muestra el valor de la traza
        a(j + 1) = i + j
        MsgBox ("a(" + Str(j + 1) + ") = " + Str(a(j + 1)))       'Muestra el valor de la traza
        suma = suma + a(i) + a(j) + b
        MsgBox ("Suma = " + Str(suma))          'Muestra el valor de la traza
        Next j
    Next i
End Sub
```

b) Transformar el algoritmo del apartado anterior en diagrama de N - S. **(1 punto)**



P3. Diseñar un algoritmo en lenguaje natural que permita aceptar los elementos de una matriz (A) de enteros de **m** filas x **n** columnas y que calcule la suma de cada una de sus filas y columnas, mostrando por pantalla dichos resultados en 2 vectores, uno de la suma de las filas vector (F) y otro de la suma de las columnas vector (C). Como estructuras de repetición no se podrán usar la de condición final ni la de condición inicial. Deberán disponerse todos los filtros necesarios para que no se produzcan errores. Realizar el algoritmo usando por lo menos las variables dadas entre paréntesis del enunciado. **(2,5 puntos)**

```
Algoritmo Suma de filas y columnas en una Matriz
variables
        entero: i (contador de filas); j (contador de columnas); m y n (dimensiones de la matriz)
          A(i, j) (matriz de m x n); F(i) (suma de elementos de filas); C(j) (suma de elementos de columnas)
comienzo
        mostrar "Introduzca las dimensiones m y n de la matriz A:"
        aceptar m , n
        si m < 0 entonces
                mostrar "Las dimensiones de la matriz no pueden ser negativas, se corrige el valor negativo de m"
                hacer m ← m * (-1)
        fin-si
        si n < 0 entonces
                mostrar "Las dimensiones de la matriz no pueden ser negativas, se corrige el valor negativo de n"
                hacer n ← n * (-1)
        fin-si
        para i ← 1 hasta m
                para j ← 1 hasta n
                        aceptar A(i, j)
                siguiente j
        siguiente i
        para i ←1 hasta m
                para j ← 1 hasta n
                        hacer F(i) ← F(i) + A(i, j)
                siguiente j
                escribir "Fila " + i + ": " + F(i)
        siguiente i
        para j ← 1 hasta n
                para i ← 1 hasta m
                        hacer C(j) ← C(j) + A(i, j)
                siguiente i
                escribir "Columna " + j + ": " + C(j)
        siguiente j
fin
```

a) Realizar la traza para f = 3 ; c = 2 con A= $\begin{pmatrix} 1 & 3 \\ 4 & 0 \\ 3 & -1 \end{pmatrix}$. **(1 punto)**

m	n	i → m	j→n	A(i, j)	i → m	j→n	F(i)	j→n	i → m	C(j)	Suma de Filas	Suma de columnas
3	2	1→3	1→2	1								
			2	3								
		2	1→2	4								
			2	0								
		3	1→2	3								
			2	-1								
					1→3	1→2	1					
						2	4				**4**	
					2	1→2	4					
						2	4				**4**	
					3	1→2	3					
						2	2				**2**	
								1→2	1→3	1		
									2	5		
									3	8		**8**
								2	1→3	3		
									2	3		
									3	2		**2**

Código en Visual Basic del algoritmo anterior:
Con un formulario sin objetos, copiar el código en la ventana ‘Ver código’ del Visual Basic.

```
Dim m As Integer    'Dimensión de las filas de la matriz A
Dim n As Integer    'Dimensión de las columnas de la matriz A
Dim A() As Integer  'Matriz de dimensiones m x n
Dim i As Integer    'Contador de filas
Dim j As Integer    'Contador de columnas
Dim F() As Integer  'Vector suma de filas
Dim C() As Integer  'Vector suma de columnas

Private Sub Form_Load()
    m = Val(InputBox("Introduzca el número de filas de la matriz A:"))
    n = Val(InputBox("Introduzca el número de columnas de la matriz A:"))
    ReDim A(m, n)
    For i = 1 To m
        For j = 1 To n
            A(i, j) = InputBox("Elemento (" & i & ", " & j & ")=")
        Next j
    Next i
    ReDim F(m)
    ReDim C(n)
    For i = 1 To m
        For j = 1 To n
            F(i) = F(i) + A(i, j)
        Next j
        MsgBox "Fila " & i & ": " & F(i)
    Next i
    For j = 1 To n
        For i = 1 To m
            C(j) = C(j) + A(i, j)
        Next i
        MsgBox "Columna " & j & ": " & C(j)
    Next j
End Sub
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO DICIEMBRE 2.010 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. Ejercicio teórico.

a) ¿De qué clase pueden ser las variables que intervienen en algoritmos que usan **subalgoritmos**?. **(1 punto)** Define estas clases. **(1 punto)**

Las variables que intervienen en un algoritmo con subalgoritmos pueden ser de dos clases: variables locales y variables globales.

- **Las variables locales: son las que se utilizan en la definición de un subalgoritmo. Sólo tienen actuación dentro del subalgoritmo en el cual han sido declaradas y no son conocidas fuera de él.**
- **Las variables globales: están definidas en el algoritmo principal y tienen actuación tanto en el algoritmo principal como en los subalgoritmos que dependen de él. Su uso es poco recomendable en ocasiones por el 'Efecto lateral'.**

b) Dado el siguiente algoritmo

```
Algoritmo Principal
variables
        entero: A; B; C
comienzo
    hacer A ← 2
    hacer B ← 3
    hacer C ← EfectoLateral (B)
    escribir A, B, C
    hacer C ← EfectoLateral (B)
    escribir A, B, C
fin
```

```
función EfectoLateral (X: entero): entero
comienzo
        hacer B ← X + 1
        EfectoLateral ← 2 * B
fin-función
```

Se pide:

Indicar por medio de una traza cuál de las siguientes soluciones sería la salida de datos de la última instrucción 'escribir' que se ejecuta, ¿por qué?. **(2 puntos)**

☐ a. 2, 3, 18

☐ b. 2, 3, 8

☐ c. **2, 5, 10**

☐ d. 2, 4, 8

Este es un ejemplo de lo que sucede si una variable global (B) modifica su valor dentro de un subalgoritmo, es lo que se conoce como 'Efecto lateral'. Al efectuar la traza se obtiene sucesivamente:

A	B	C	escribir	*C*	escribir
2	3	8	2, 4, 8	10	2, 5, 10

X	B	*EfectoLateral*
3	4	8
4	5	10

P2. Diseñar un algoritmo en pseudocódigo que permita cargar una matriz (U) en memoria y comprobar si la misma es unitaria o no, ¡ojo! abandonando la comprobación cuando algún elemento no sea de la matriz unitaria. Una matriz unitaria de orden (n) es aquella que tienen n filas (f) y n columnas (c), con todas sus componentes a 0 excepto la diagonal principal que está a 1.

$$U = \begin{array}{c} \\ f \\ \\ \\ \\ \end{array} \begin{array}{c} \overset{c}{} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}_{n \times n} \end{array}$$

Importante: la única instrucción de repetición que podrá ser usada es la estructura de repetición con condición inicial. Deberán disponerse todos los filtros necesarios para que no se produzcan errores. Diseñar el algoritmo usando por lo menos las variables dadas entre paréntesis del enunciado. **(2 puntos)**

```
Algoritmo Matriz unitaria
variables
        entero: n (filas y columnas de la matriz U); f, c (contadores de filas y columnas de la matriz U); U (f, c) (matriz U de
                dimensiones f, c)
        alfanumérico: sw  (testigo)
comienzo
        //Entrada de la dimensión de la matriz
        mostrar "Indique la dimensión de la matriz unitaria a introducir:"
        aceptar n
        mientras n <= 0
                mostrar "La dimensión de la matriz unitario no puede ser menor o igual que cero, vuelva a introducir la dimensión"
                aceptar n
        fin-mientras
        //Entrada de los elementos de la matriz
        hacer f ← 1
        mientras f <= n
                hacer c ← 1
                mientras c <= n
                        mostrar "Introduce el valor del elemento de la matriz U(" + f + ", " + c + ")"
                        aceptar U(f, c)
                        hacer c ← c + 1
                fin-mientras
                hacer f ← f + 1
        fin-mientras
        //Comprobación de los elementos de la matriz
        si n = 1 y U(1, 1) = 1 entonces
                mostrar "La matriz es unitaria"
        si-no
                hacer f ← 1
                hacer sw ← "cierto"
                mientras f <= n y sw = "cierto"     // Comprobación del testigo sw para no seguir comprobando
                        hacer c ← 1
                        mientras c <=  n y sw = "cierto"  // Comprobación del testigo sw para no seguir comprobando
                                si f = c y U(f, c) <> 1 entonces
                                        hacer sw ← "falso"  // Testigo para no seguir comprobando
                                sino
                                        si f <> c y U(f, c) <> 0 entonces
                                                hacer sw ← "falso"  // Testigo para no seguir comprobando
                                        fin-si
                                        hacer c ← c + 1   // Contador para salir del bucle mientras
                                fin-si
                        fin-mientras
                        hacer f ← f + 1       // Contador para salir del bucle mientras
                fin-mientras
                si sw = "cierto"  entonces
                        mostrar "La matriz es unitaria"
                si-no
                        mostrar "La matriz no es unitaria"
                fin-si
        fin-si
fin
```

Ejemplo en Visual Basic para comprobar:

Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Option Explicit

Private Sub Form_Load()
    Dim n As Integer       'número de filas y columnas de la matriz U
    Dim f As Integer       'contador de filas de la matriz U
    Dim c As Integer       'contador de columnas de la matriz U
    Dim U() As Integer     'matriz A de dimensiones f, c
    Dim sw As String       'interruptor para abandonar la comprobación de matriz unitaria

    'Entrada de la dimensión de la matriz
```

```
    n = Val(InputBox("Indique la dimensión de la matriz unitaria a introducir:", "Entrada de datos"))
    While n <= 0
            n = Val(InputBox("La dimensión de la matriz unitario no puede ser menor o igual que cero, vuelva a introducir la
dimensión", "Entrada de datos"))
    Wend
    'Entrada de los elementos de la matriz
    ReDim U(n, n)     'Se indica en este momento las dimensiones y el número de elementos de la matriz U
    f = 1
    While f <= n
        c = 1
        While c <= n
            U(f, c) = Val(InputBox("Introduce el valor del elemento de la matriz U(" & f & ", " & c & ")", "Entrada de datos"))
            c = c + 1
        Wend
        f = f + 1
    Wend
    'Comprobación de los elementos de la matriz
    If n = 1 And U(1, 1) = 1 Then
        MsgBox "La matriz es unitaria"
    Else
        f = 1
        sw = "verdadero"
        While f <= n And sw = "cierto"          'Comprobación del testigo sw para no seguir comprobando
            c = 1
            While c <= n And sw = "cierto"          ' Comprobación del testigo sw para no seguir comprobando
                If f = c And U(f, c) <> 1 Then
                    sw = "falso"          'Testigo para no seguir comprobando
                Else
                    If f <> c And U(f, c) <> 0 Then
                        sw = "falso"          'Testigo para no seguir comprobando
                    End If
                    c = c + 1         'Contador para salir del bucle mientras
                End If
            Wend
            f = f + 1         'Contador para salir del bucle mientras
        Wend
        If sw = "cierto" Then
            MsgBox "La matriz es unitaria"
        Else
            MsgBox "La matriz no es unitaria"
        End If
    End If
End Sub
```

a) Realiza la traza para los siguientes casos. **(1 punto)**

a) $U = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ b) $U = \begin{pmatrix} 1 \end{pmatrix}$

	n	f	c	U(f, c)	c	f	solución	f	sw	c	c	sw	f	solución
a)	3	1	1	U(1, 1) = 1	2									
				U(1, 2) = 0	3									
				U(1, 3) = 0	4	2								
			1	U(2, 1) = 0	2									
				U(2, 2) = 1	3									
				U(2, 3) = 0	4	3								
			1	U(3, 1) = 0	2									
				U(3, 2) = 0	3									
				U(3, 3) = 0	4	4		1	verdadero	1	2			
											3			
											4		2	
										1	2			
											3			
											4		3	
										1	2			
											3	falso		
											4		4	**No unitaria**

b)	1	1	1	U(1, 1) = 1	2	2	**Unitaria**							

P3. ¿Sería correcto el siguiente subalgoritmo si lo que se desea es obtener un vector F con la sucesión de Fibonacci F(0, 1, 1, 2, 3, 5, 8, 13, 21, 34,…) para n términos y la suma de sus términos?. Justifique la respuesta por medio de una traza. **(2 puntos)**

```
procedimiento fibonacci (n = entero)
variables
        entero: i (contador de elementos del vector F); F(n) (Sucesión de Fibonacci); sumaFibonacci (Suma de los elementos de
        la sucesión de Fibonacci)
        alfanumérico: elem (Acumulador de elementos del vector F(n); vectorFibonacci (Solución mostrando el vector F(n)
comienzo
        si n = 1 entonces
                hacer   F(1) ← 0
                        sumaFibonacci ← 0
        sino
        hacer       F(1) ← 0
                    F(2) ← 1
                    sumaFibonacci ← 1
                si n > 2 entonces
                    para i ← 3 hasta n
                        hacer   F(i) ← F(i - 1) + F(i - 2)
                                sumaFibonacci ← sumaFibonacci + F(i)
                    siguiente i
                fin – si
        fin – si
        si n > 1 entonces
                para i ← 2 hasta n
                    hacer   elem ← F(i)
                    si i < n entonces
                        hacer   elem ← elem + ", "
                    fin – si
                    hacer   vectorFibonacci ← vectorFibonacci + elem
                siguiente i
        fin – si
        si n = 1 entonces
                mostrar "La sucesión de Fibonacci para un valor de n = " + n + " es F(0), cuya suma vale " + sumaFibonacci
        sino
                hacer   vectorFibonacci ← "0, " + vectorFibonacci
                        mostrar "La sucesión de Fibonacci para un valor de n = " + n + " es F(" + vectorFibonacci + "), cuya suma
                        vale " + sumaFibonacci
        fin – si

fin-procedimiento
```

Es correcto. Si realizamos la comprobación por medio de la traza para n = 4, n = 2 y n = 1 obtenemos los siguientes valores:

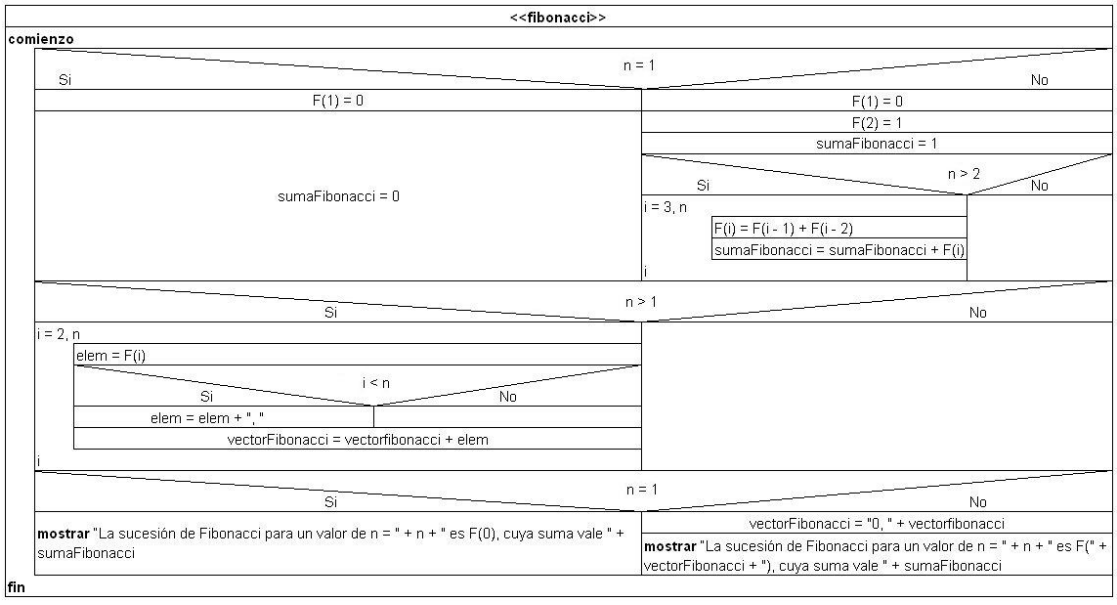
n	F(1)	F(2)	sumaFibonacci	i	F(i)	sumaFibonacci	i	elem	elem	vectorFibonacci	vectorFibonacci
4	0	1	1	3 → 4	F(3) = 1	2					
				4	F(4) = 2	**4**					
							2 → 4	1	1,	1,	
							3	1	1,	1, 1,	
							4	2	2	1, 1, 2	**0, 1, 1, 2**
2	0	1	1				2 → 2	1	1	1	**0, 1**
1	0		0								**0**

Ejemplo en Visual Basic para comprobar:
Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Option Explicit
Private Sub Form_Load()
    'Declaración de variables globales
    Dim numElem As Integer      'Nº de elementos de la sucesión de Fibonacci
    'Entrada de datos
    Do
        numElem = Val(InputBox("Introduzca el nº de elementos de la sucesión de Fibonacci:", "Entrada de datos"))
        If numElem < 1 Then
```

```
            MsgBox "Para obtener la sucesión de Fibonacci debe ser un nº positivo mayor que 0"
        End If
    Loop Until numElem > 0

    'Salida de resultados
    Fibonacci (numElem)
End Sub

Sub Fibonacci(n As Integer)
    'Declaración de variables locales
    Dim i As Integer        'Contador de índices del vector F
    Dim F() As Integer      'Vector con los elementos de la sucesión de Fibonacci
    Dim elem As String      'Variable tipo acumulador de elementos del vector Fibonacci(n)
    Dim vectorFibonacci As String       'Solución mostrando el vector Fibonacci(n)
    Dim sumaFibonacci As Integer        'Suma de las componentes de la sucesión de Fibonacci

    'Cálculo de la sucesión de Fibonacci
    ReDim F(n)        'Se indica en este momento la dimensión y el número de elementos de la vector F
    If n = 1 Then
        F(1) = 0
        sumaFibonacci = 0
    Else
        F(1) = 0
        F(2) = 1
        sumaFibonacci = 1
        If n > 2 Then
            For i = 3 To n
                F(i) = F(i - 1) + F(i - 2)
                sumaFibonacci = sumaFibonacci + F(i)
            Next i
        End If
    End If

    'Salida de resultados
    If n > 1 Then
        For i = 2 To n
            elem = F(i)
            If i < n Then
                elem = elem & ", "      'Después de la coma hay un espacio
            End If
          vectorFibonacci = vectorFibonacci & elem
        Next i
    End If
    If n = 1 Then
        MsgBox "La sucesión de Fibonacci para un valor de n = " & n & " es F(0), cuya suma vale " & sumaFibonacci
    Else
        vectorFibonacci = "0, " & vectorFibonacci
        MsgBox "La sucesión de Fibonacci para un valor de n = " & n & " es F(" & vectorFibonacci & "), cuya suma vale " &
sumaFibonacci
    End If
End Sub
```

a) Transformar a diagrama de Chapin. **(1 punto)**



Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JULIO 2010 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. Ejercicio teórico.

a) Indica las reglas para la construcción de identificadores. **(1,25 puntos)**

- **Debe resultar significativo, sugiriendo lo que representa.**
- **No podrá coincidir con palabras reservadas, propias del lenguaje algorítmico o de programación.**
- **Se admitirá un máximo de 32 caracteres.**
- **Comenzará siempre por un carácter alfabético y los siguientes podrán ser letras, dígitos o el símbolo de subrayado (underscore).**
- **Podrá ser utilizado indistintamente escrito en mayúsculas o en minúsculas.**

b) Calcula $A3_{16}$^100_2 = X_8 donde X es el valor de la expresión en octal. **(1 puntos)**

Aplicando el teorema fundamental de la numeración a cada término tenemos:

$A3_{16 \rightarrow 10} = 10 * 16^1 + 3 * 16^0 = 163_{10}$

$100_{2 \rightarrow 10} = 1 * 2^2 + 0 * 2^1 + 0 * 2^0 = 4_{10}$

$X_{10} = 163^4 = 705911761_{10}$

Si pasamos a octal 705911761, tenemos:

$705911761_{10 \rightarrow 8} = X_8$

División	Cociente entero	Resto
705911761/8	88238970	1
88238970/8	11029871	2
11029871/8	1378733	7
1378733/8	172341	5
172341/8	21542	5
21542/8	2692	6
2692/8	336	4
336/8	42	0
42/8	5	2
5/8	0	5

$X_8 = 5204655721_8$

Comprobación $1 * 8^0 + 2 * 8^1 + 7 * 8^2 + 5 * 8^3 + 5 * 8^4 + 6 * 8^5 + 4 * 8^6 + 0 * 8^7 + 2 * 8^8 + 5 * 8^9 = 1 + 16 + 448 + 2560 + 20480 + 196608 + 1048576 + 0 + 33554432 + 671088640 = 705911761_{10}$ c.q.d

c) Calcula las siguientes expresiones lógicas indicando paso a paso el orden de prioridad de los operadores: **(0,75 puntos)**

- No(4 > 6) y 2 * redondeo(2,5) > 6

No(falso) y 2 * redondeo(2,5) > 6
No(falso) y 2 * 3 > 6;
verdadero y 2 * 3 > 6;
verdadero y 6 > 6;
verdadero y falso;
falso

- abs(-(2 – trunc(2,9) + 0,5)) < 0,5

abs(- (2 – 2 + 0,5)) < 0,5;
abs(- (0 + 0,5)) < 0,5;
abs(- (0,5)) < 0,5;
abs(- 0,5) < 0,5;
0,5 < 0,5;
falso

- 3 + 2^2 –3 * ent(redondeo(0,9) / 2) < 6 o falso

3 + 2^2 – 3 * ent(1 / 2) < 6 o falso;
3 + 2^2 – 3 * ent(0,5) < 6 o falso;
3 + 2^2 – 3 * 0 < 6 o falso;
3 + 4 – 3 * 0 < 6 o falso;
3 + 4 – 0 < 6 o falso;
7 < 6 o falso;
falso o falso;
falso

d) Dadas las siguientes estructuras de repetición:

```
Algoritmo Uno
variables  entero: n; c; i
comienzo
      hacer i ← 3
      mientras i > 0
            hacer c ← c + 1
            hacer i ← i - 1
      fin - mientras
      mostrar c
fin
```

```
Algoritmo Dos
variables  entero: n; c; i
comienzo
      hacer i ← 1
      mientras i <= 3
            hacer c ← c + 1
            hacer i ← i + 1
      fin - mientras
      mostrar c
fin
```

```
Algoritmo Tres
variables  entero: n; c; i
comienzo
      repetir
            hacer c ← c + 1
            hacer i ← i + 1
      hasta i = 3
      mostrar c
fin
```

```
Algoritmo Cuatro
variables  entero: n; c; i
comienzo
      hacer i ← 1
      repetir
            hacer c ← c + 1
            hacer i ← i + 1
      hasta i > 3
      mostrar c
fin
```

```
Algoritmo Cinco
variables  entero: n; c; i
comienzo
      hacer i ← 3
      repetir
            hacer c ← c + 1
            hacer i ← i - 1
      hasta i = 0
      mostrar c
fin
```

```
Algoritmo Seis
variables  entero: n; c; i
comienzo
      hacer i ← 3
      repetir
            hacer c ← c + 1
            hacer i ← i - 1
      hasta i < 1
      mostrar c
fin
```

d1. ¿Son equivalentes todas estas estructuras de repetición o hay alguna que no sea equivalente?. **(0,25 puntos)**, Justifica las estructuras equivalentes indicando el por qué y realizando su traza. **(0,25 puntos)**

Traza algoritmo uno		
i	**c**	**i**
3	1	2
	2	1
	3	0

Traza algoritmo dos		
i	**c**	**i**
1	1	2
	2	3
	3	4

Traza algoritmo tres		
	c	**i**
	1	1
	2	2
	3	3

Traza algoritmo cuatro		
i	**c**	**i**
1	1	2
	2	3
	3	4

Traza algoritmo cinco		
i	**c**	**i**
3	1	2
	2	1
	3	0

Traza algoritmo seis		
i	**c**	**i**
3	1	2
	2	1
	3	0

Sí, todas son semejantes puesto que al realizar la traza a cada una de ellas se obtiene el mismo resultado para c, el valor 3, que es el que muestra la instrucción 'mostrar'.

d2. Si es posible escribe otra estructura de repetición con condición inicial equivalente a las dadas en el apartado anterior sin inicializar ninguna variable. En caso de ser posible justifica la solución dada por medio de su traza. **(0,5 puntos)**

Por ejemplo:

```
Algoritmo Otra solución
variables  entero: n; c; i
comienzo
        mientras i < 3
                hacer c ← c + 1
                hacer i ← i + 1
        fin - mientras
        mostrar c
fin
```

Traza algoritmo d2		
i	**c**	**i**
0	1	1
	2	2
	3	3

P2. Dada la siguiente función recursiva:

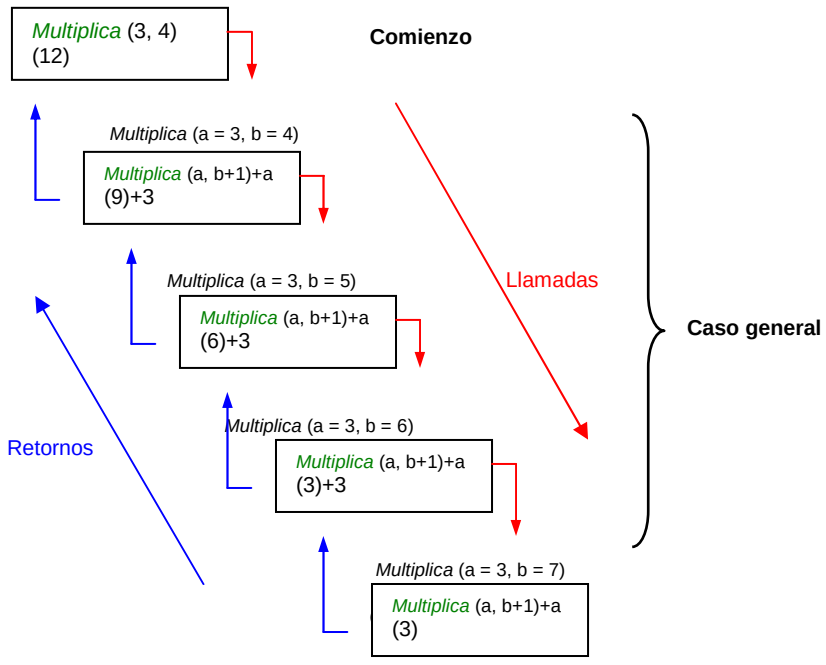
```
Función  Multiplica (a,b: entero): entero
comienzo
    si b = 7 entonces
        hacer Multiplica  ← a
    si-no
        hacer Multiplica ← Multiplica (a, b + 1) + a
    fin-si
fin-función
```

a) ¿Qué valor devolverá la función con a = 3 y b = 4? **(0,5 puntos)**
Multiplica = 12

b) Realiza las llamadas y retornos para a = 3 y b = 4.**(2 puntos)**

La recursividad usa la pila del computador. Una pila es una estructura de datos lineal en la cual sólo se accede a uno de sus extremos, llamado *cima*, tanto para insertar como para extraer datos.

	a	b		Multiplica
4º	3	7 1º	entonces	3 + = 3
3º	3	6 2º	entonces	3 + 3 + = 6
2º	3	5 3º	entonces	3 + 3 + 3 + = 9
1º	3	4 4º	entonces	3 + 3 + 3 + 3 = 12

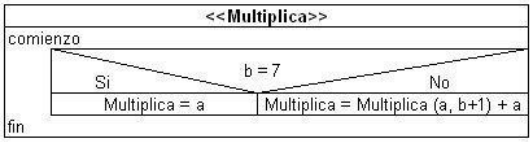


Comprobación realizada con Visual Basic:
Con un formulario vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Option Explicit
Private Sub Form_Load()
  Dim x As Integer
  Dim y As Integer
  Dim z As Integer
  x = 3
  y = 4
  z = Multiplica(x, y)
```

```
    MsgBox "El valor devuelto por la función Multiplica es: " & Format(z)
End Sub

Function Multiplica(a, b As Integer) As Integer
    If b = 7 Then   'caso base
        MsgBox "Valor de llamada a a: " & Format(a) & vbCrLf & "Valor de llamada a b: " & Format(b)
        Multiplica = a
    Else
        MsgBox "Valor de llamada a a: " & Format(a) & vbCrLf & "Valor de llamada a b: " & Format(b)
        Multiplica = Multiplica(a, b + 1) + a    'caso general o recursivo
    End If
    MsgBox "Valor de retorno a: " & Format(a) & vbCrLf & "Valor de retorno b: " & Format(b) & vbCrLf & "Valor de retorno
            Multiplica: " & Format(Multiplica)
End Function
```

c) Transformar en diagrama N – S la función recursiva anterior. **(0,5 puntos)**



P3. Una matriz A de la forma:

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots\ldots & a_{1n} \\ a_{21} & a_{22} & \ldots\ldots & a_{2n} \\ . & . & . & \\ . & . & . & \\ . & . & . & \\ a_{n1} & a_{n2} & \ldots\ldots & a_{nn} \end{pmatrix}_{n \times n}$$

es simétrica si se cumple que A (f, c) = A (c, f) para 1 < = f < = n y 1 < c < = n. El algoritmo adjunto lee una matriz de n x n y determina si es o no simétrica.

```
01    Algoritmo ¿Matriz simétrica?
02    variables
03          alfanumérico: n (filas y columnas de la matriz A); f, c (contadores); A (f, c) (matriz A de dimensiones f, c); no (contador de
            elementos no simétricos)
04    comienzo
05      aceptar n
06      mientras n < 0
07          aceptar n
08      fin-mientras
09      para f ← 1 hasta n
10          para c ← 1 hasta n
11              aceptar A()
12          siguiente
13      siguiente
14      si n = 1 entonces
15          mostrar "La matriz es simétrica"
16      si-no
17          hacer f ← 2
18          mientras f ≤ n
19              hacer c ← 1
20              mientras c ≤  f – 1
21                  hacer
22                  si A(f, c) <> A(c, f) entonces
23                      mostrar "El elem. A(" + f + ", " + c + ") es distinto al elem. A(" + c + ", " + f +")"
24                      hacer no ← no + 1
25                  fin-si
26                  hacer c ← c + 1   // Contador para salir del bucle mientras
27              fin-mientras
28              hacer f ← 1 + f      // Contador para salir del bucle mientras
29          fin-mientras
30          si no = 0  entonces
31              mostrar "La matriz es simétrica"
32          si-no
33              mostrar "La matriz no es simétrica"
34          fin-si
35      fin-si
36  fin
```

a) Del algoritmo anterior, identifica la línea o líneas de pseudocódigo que contienen errores de sintaxis algorítmica e indica la corrección a aplicar para que el algoritmo pueda funcionar al realizar su traza. **(1,5 puntos)**

- Línea 03: es incorrecta el tipo de declaración de variables, la forma correcta es de tipo entero excepto la variable A(f,c) puesto que debería ser de tipo real (dentro de los reales también están los enteros) o de tipo alfanumérico si se van a tener caracteres no numéricos.
- Línea 03: es incorrecta el nombre de la variable 'no', coincide con el operador lógico 'negación lógica'
- Línea 06: la comprobación no es correcta del todo, la forma correcta es n <= 0
- Línea 11: es incorrecto llamar a la variable tipo matriz de esa forma, la forma correcta es A(f, c)
- Línea 12: es incorrecto el pie del bloque de control 'para', falta indicar la variable para contar, la forma correcta es 'siguiente c'
- Línea 13: es incorrecto el pie del bloque de control 'para', falta indicar la variable para contar, la forma correcta es 'siguiente f'
- Línea 18: es incorrecto el operador relacional o lógico, su forma correcta es <=
- Línea 20: es incorrecto el operador relacional o lógico, su forma correcta es <=
- Línea 21: es incorrecto el uso de la instrucción 'hacer' sin proceso

b) Diseña un refinamiento del algoritmo anterior que optimice el funcionamiento del mismo, es decir, que se consiga hacer más eficiente el algoritmo. Por eficiencia se entiende la aptitud, capacidad para llevar a cabo un trabajo o una tarea con el mínimo gasto de recursos, tiempo, y pasos, y con el máximo provecho o rendimiento del algoritmo. **(1,5 puntos)**

Por ejemplo podemos declarar una variable de nombre "sw" y de tipo alfanumérica, entera o lógica, como queramos, para usarla como un switch (interruptor, conmutador, testigo, etc) que nos permita tener una señal para abandonar un bucle de repetición y analizar el estado de esa variable para tomar decisiones. Incluso serviría el contador no para analizar si es mayor a cero sino para interrumpir el bucle de repetición.

```
Algoritmo ¿Matriz simétrica?
variables
        entero: n (filas y columnas de la matriz A); f, c (contadores)
        real: A (f, c) (matriz A de dimensiones f, c) // Podría ser de tipo alfanumérico si se van a tener letras
        alfanumérico: sw  (testigo)
comienzo
        aceptar n
        mientras n <= 0
            aceptar n
        fin-mientras
        para f ← 1 hasta n
                para c = 1 hasta n
                    aceptar A(f, c)
                siguiente c
        siguiente f
        si n = 1 entonces
            mostrar "La matriz es simétrica"
        si-no
            hacer sw ← "verdad"
            hacer f ← 2    // Para comenzar las comparaciones por el elemento (2, 1) con el (1, 2)
            mientras f <= n y sw = "verdad"   // Comprobación del testigo sw para no seguir comprobando
                hacer c ← 1
                mientras c <=  f – 1 y sw = "verdad"  // Comprobación del testigo sw para no seguir comprobando
                    si A(f, c) <> A(c, f) entonces
                        hacer sw ← "falso"  // Testigo para no seguir comprobando
                    fin-si
                    hacer c ← c + 1   // Contador para salir del bucle mientras
                fin-mientras
                hacer f ← f + 1       // Contador para salir del bucle mientras
            fin-mientras
            si sw = "verdad"  entonces
                mostrar "Simétrica"
            si-no
                mostrar "No simétrica"
            fin-si
        fin-si
fin
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JUNIO 2.010 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. Ejercicio teórico.

a) Indica y describe los tipos de memorias usadas en los computadores. **(1 punto)**

a. **Registros: son memorias elementales para que las unidades de la CPU puedan almacenar datos o instrucciones temporalmente.**
b. **Memoria intermedia: es una memoria buffer o memoria que se utiliza como regulador y sistema de almacenamiento intermedio entre dispositivos de un sistema informático.**
c. **Memoria interna: es la memoria principal o central (CM) de la computadora.**
d. **Memoria auxiliar: es una memoria secundaria o la representada por un periférico de almacenamiento.**
e. **Memoria virtual: es un tipo de memoria interna y una parte de una memoria rápida.**

b) Sistema de numeración:

a. ¿Cuál es el número máximo positivo representable con 8 bits en decimal?. ¿Por qué? **(0,5 puntos)**
Si recordamos, un bit es la unidad mínima de información, sus valores pueden ser 0 u 1. Entonces el nº máximo positivo representable con 8 bits es el $11111111_2 = 255$

b. Clasifica los sistemas de numeración y defínelos. **(0,5 puntos)**
Los sistemas de numeración pueden clasificarse en dos grandes grupos: posicionales y no-posicionales.
En los sistemas no-posicionales los dígitos tienen el valor del símbolo utilizado, que no depende de la posición (columna) que ocupan en el número.
En los sistemas de numeración ponderados o posicionales el valor de un dígito depende tanto del símbolo utilizado, como de la posición que ése símbolo ocupa en el número.

c. Define el teorema fundamental de la numeración. **(1 punto)**
Este teorema establece la forma general de construir números en un sistema de numeración posicional. Primero estableceremos unas definiciones básicas:
N**, número válido en el sistema de numeración.**
b**, base del sistema de numeración. Número de símbolos permitidos en el sistema.**
d_i**, un símbolo cualquiera de los permitidos en el sistema de numeración.**
n**,: número de dígitos de la parte entera.**
, coma fraccionaria. Símbolo utilizado para separar la parte entera de un número de su parte fraccionaria.
k**,: número de dígitos de la parte decimal.**
La fórmula general para construir un número *N*, con un número finito de decimales, en un sistema de numeración posicional de base *b* es la siguiente:

$$N = \begin{cases} \langle d_{(n-1)} \dots d_1 d_0, d_{-1} \dots d_{-k} \rangle = \sum_{i=-k}^{n} d_i b^i \\ N = d_n b^n + \dots + d_1 b^1 + d_0 b^0 + d_{-1} b^{-1} + \dots + d_{-k} b^{-k} \end{cases}$$

El valor total del número será la suma de cada dígito multiplicado por la potencia de la base correspondiente a la posición que ocupa en el número.

c) Contesta las siguientes preguntas.

1) ¿Es obligatorio que una función use parámetros actuales en la llamada o ficticios en la declaración?, ¿Por qué?. **(0,5 puntos)**

No es obligatorio. Una la función puede trabajar sin pasar parámetros a la declaración puesto que es autónoma ya que devuelve datos através de su nombre.

2) Escribe un algoritmo sencillo en pseudocódigo que use una función y que ésta no emplee parámetros (debe aparecer la llamada y su declaración). **(0,5 puntos)**

```
algoritmo Ejemplo de uso función sin parámetros
variables
    entero: a (variable auxiliar)
comienzo
    hacer a ← numero()
    si a = 1 entonces
        mostrar "Se ha pulsado la tecla 1"
    sino
        mostrar "Se ha pulsado una tecla distinta a la del 1"
    fin-si
fin

Función numero(): entero
Variables
    entero: tecla(tecla pulsada)
    mostrar "Teclee un 1 o cualquier otra tecla"
    aceptar tecla
    hacer numero ← tecla
fin-función
```

Ejemplo en Visual Basic para comprobar:

Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Dim a As Integer
Private Sub Form_Load()
    a = numero()        'Llamada a la función sin pasar parámetros
    If a = 1 Then
        MsgBox "Se ha pulsado la tecla 1"
    Else
        MsgBox "Se ha pulsado una tecla distinta a la del 1"
    End If
End Sub

Function numero() As Integer        'Declaración de la función sin usar parámetros
    numero = Val(InputBox("Teclee un 1 o cualquier otra tecla ", "Entrada de datos"))
End Function
```

P2. Realizar la traza del siguiente algoritmo, indicando los valores que toman las variables i, j, a(i), a(j), a(j +1), b y suma. **(1,5 puntos)**

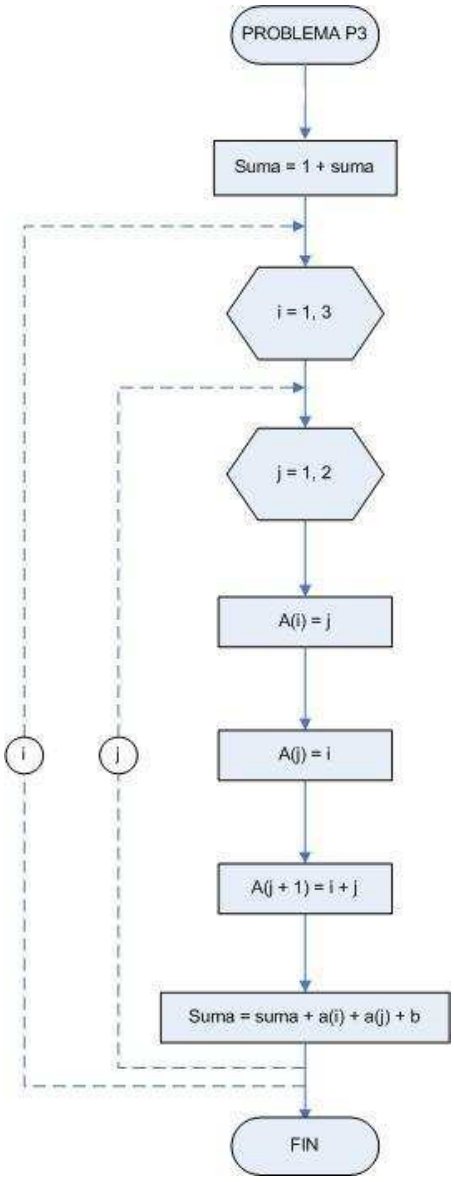
```
Algoritmo Traza
variables
        entero: a(i), a(j), a(j +1); i , j; suma; b      // No se ha indicado la descripción por no ser necesaria
comienzo
        hacer suma ← 1 + suma
        para i ← 1 hasta 3
            para j ← 1 hasta 2
                hacer    a(i) ← j
                         a(j) ← i
                         a(j +1) ← i + j
                         suma ← suma + a(i) + a(j) + b
            siguiente j
        siguiente i
fin
```

¡Ojo¡, la traza ha de hacerse como se ha explicado en clase o como las tenéis en la colección de ejercicios resueltos, no hay que inventar o innovar.

suma	i	j	a(i)	a(j)	a(j +1)	suma
1	1 → 3	1 → 2	a(1) = 1	a(1) = 1	a(2) = 2	3
		2	a(1) = 2	a(2) = 1	a(3) = 3	6
	2	1 → 2	a(2) = 1	a(1) = 2	a(2) = 3	11
		2	a(2) = 2	a(2) = 2	a(3) = 4	15
	3	1 → 2	a(3) = 1	a(1) = 3	a(2) = 4	19
		2	a(3) = 2	a(2) = 3	a(3) = 5	27

Una vez que se declara una variable, el sistema le asigna un valor por defecto, ya que la variable está creada y además con su tipo de dato. Así, que si se usa la variable 'suma' o 'b', ésta contiene en memoria un dato, que está claro que si uno no lo inicializa al valor que le interesa contendrá el de defecto, en este caso suma = 0 y b = 0. No es normal usar el valor por defecto, ya que casi siempre se necesita si la variable es de tipo entero que sea distinto a cero.

a) Transformar el algoritmo del apartado anterior en diagrama de flujo. **(1 puntos)**



P3. Realizar un algoritmo en lenguaje natural que permita aceptar los elementos de un vector (V) de (n) componentes y los escriba al revés mostrándolos como en el ejemplo adjunto, con sus paréntesis y comas, sin utilizar otro vector auxiliar. Como estructura de repetición únicamente se podrá usar la de condición final. Deberán disponerse todos los filtros necesarios para que no se produzcan errores. Diseñar el algoritmo usando por lo menos las variables dadas entre paréntesis del enunciado. **(2,5 puntos)**

Ejemplo:

Vector leído: V(1, 2, 3, 4, e)
Vector escrito: *V(e, 4, 3, 2, 1)* La parte en cursiva es lo que tiene que escribirse o mostrarse en la pantalla.

```
Algoritmo Invertir vector
variables
    entero: n (Nº de elementos del vector V(n); i (Contador de elementos del vector V(n))
    alfanumérico: V(n) (Vector de n componentes); elem (Acumulador de elementos del V(n));
    vectorInvertido (Solución mostrando el vector V(n) invertido)
comienzo
    // Entrada de datos
    repetir
        mostrar "Introduzca el nº de elementos del vector a invertir:"
        aceptar n
        si n < 1 entonces
            mostrar "El nº de elementos debe ser un nº posito y mayor que 0"
        fin-si
    hasta n > 0
    hacer i ← 1
    repetir
        mostrar "Introduzca el valor del elemento " + i + " del vector a invertir:"
        aceptar V(i)
        hacer i ← i + 1
    hasta i > n
    // Salida de resultados
    si n = 1 entonces
        hacer i ← 1
        mostrar "V(" + V(i) + ")"
    sino
        hacer i ← n
        repetir
            hacer elem ← V(i)
            hacer elem ← elem + ", "      // después de la coma hay un espacio
            hacer vectorInvertido ← vectorInvertido + elem
            hacer i ← i - 1
        hasta i = 1
            hacer vectorInvertido ← vectorInvertido + V(1)
            mostrar "V(" + vectorInvertido + ")"
    fin-si
fin
```

Ejemplo en Visual Basic para comprobar:
Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Option Explicit
'Dado un vector invertir sus elementos sin usar un vector auxiliar
Private Sub Form_Load()
    Dim n As Integer  'Nº de elementos del vector V(n)
    Dim i As Integer  'Contador de elementos del vector V(n)
    Dim V() As String     'Vector de n componentes
    Dim elem As String      'Variable tipo acumulador de elementos del vector V(n)
    Dim vectorInvertido As String       'Solución mostrando el vector V(n) invertido

    'Entrada de datos
    Do
        n = Val(InputBox("Introduzca el nº de elementos que va a tener el vector a invertir su orden:", "Entrada de datos"))
        If n < 1 Then
            MsgBox "El nº de elementos del vector V(n) debe ser un nº positivo mayor que 0"
        End If
    Loop Until n > 0
    i = 1
    ReDim V(n)
    Do
        V(i) = InputBox("Introduzca el valor del elemento " & Str(i) & " del vector V(n):", "Entrada de datos")
        i = i + 1
    Loop Until i > n

    'Salida de resultados
    If n = 1 Then
        i = 1
        MsgBox "V(" & V(i) & ")"
    Else
        i = n
        Do
            elem = V(i)
            elem = elem & ", "        'después de la coma hay un espacio
            vectorInvertido = vectorInvertido & elem
            i = i - 1
        Loop Until i = 1
```

```
        vectorInvertido = vectorInvertido & V(1)
        MsgBox "V(" & vectorInvertido & ")"
    End If
End Sub
```

a) Realizar la traza del apartado anterior para n = 3 y V(a, b, 3) y n = 1 y V(-2). **(1 punto)**

n	i	V(i)	i	Solución	i	elem	elem	vectorInvertido	vectorInvertido	Solución
3	1	a								
	2	b								
	3	3								
					3	3	3,	3,		
					2	b	B,	3, b,		
					1					
									3, b, a	**V(3, b, a)**

1	1	-2								
	2									
			1	**V(-2)**						

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO DICIEMBRE 2.009 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

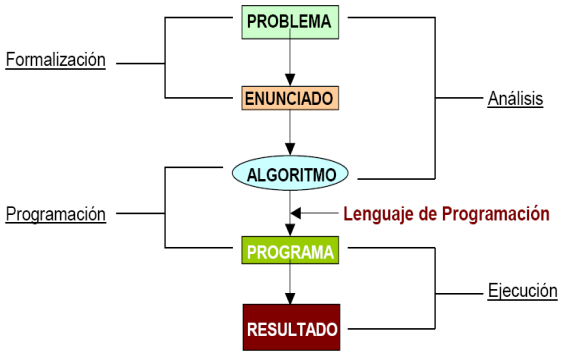
P1. Ejercicio teórico.

a) Indica y esquematiza las fases de un proceso de programación. **(1 y 0,5 puntos respectivamente)**

Fases:

a. **Identificación y formulación del problema.**
b. **Análisis del problema.**
   i. **Datos de partida (entrada de datos).**
   ii. **Datos de resultado (salida de datos).**
c. **Determinación de posibles métodos de resolución del problema.**
d. **Descripción del algoritmo adoptado.**
e. **Elección del lenguaje de programación y codificación del programa (se obtiene código fuente).**
f. **Testing (chequeo del código fuente por medio de test).**
g. **Debugging (depuración del código fuente de errores sintácticos y lógicos).**
h. **Compilación del código fuente (se obtiene código objeto).**
i. **Linkaje (programa montador o enlazador para el código objeto, se obtiene el código ejecutable EXE).**
j. **Chequeo y documentación del código ejecutable.**
   i. **Documentación interna (Ayudas en pantalla)**
   ii. **Documentación externa (Manuales)**

Esquema:



b) Definir los siguientes términos: **(1 punto)**

- Bit
- Byte
- Octeto
- Palabra

Bit: es la unidad mínima de información, sus valores podrán ser 0 u 1.
Byte: es la sucesión de dígitos binarios (Bits) adyacentes, tratados por el ordenador como una unidad de información de 8 Bits que equivalen a un carácter.
Octeto: es la información de 8 Bits que equivalen a un carácter. Si combinamos 8 números binarios hay exactamente 2^8 combinaciones, lo que da 256 posibilidades de combinación que es la cantidad de caracteres del código ASCII.
Palabra: es un conjunto de Bits, que se tratan como una sola unidad de información y que generalmente constan de 8, 16, 32 Bits...

c) Contesta a las siguientes preguntas.

1) ¿Cómo se realiza la llamada a una función desde un algoritmo en pseudocódigo?. **(0,75 puntos)** Escribe un ejemplo de llamada a función desde un algoritmo en pseudocódigo. **(0,25 puntos)**

Para llamar a una función se utiliza su nombre seguido por los parámetros actuales o reales entre paréntesis en una expresión.

Sintaxis general:

```
algoritmo nombre_algoritmo
constantes
variables   tipo de dato: nombre_de_variable1 (descripción de la variable1)
comienzo
      acción
      acción
      acción nombre_función (lista de parámetros actuales)
      acción
fin
```

Ejemplos de llamada escribir (raíz2 (x)) o hacer (f (p1, p2, p3))

2) ¿Cómo se realiza la declaración de una función en pseudocódigo?. **(0,75 puntos)** Escribe un ejemplo de declaración de función en pseudocódigo. **(0,25 puntos)**

La declaración de una función, por ser un subalgoritmo, se realiza de forma parecida a la de los algoritmos. Constan de:

- **Cabecera con el nombre de la función y entre paréntesis los parámetros formales o ficticios indicándose el tipo de dato para los parámetros por medio de dos puntos y al final el tipo de dato que devuelve la función.**
- **Cuerpo de la función. Dentro de este cuerpo estará el bloque de declaraciones y el bloque de instrucciones. Este debe incluir una instrucción mediante la que la función tomará un valor para devolverlo al algoritmo principal.**

Siempre que desde un algoritmo se hace una llamada a una función, automáticamente se establece una correspondencia entre los Parámetros formales y actuales, según la posición de cada uno de ellos, debiendo existir el mismo número de parámetros formales y actuales.

Sintaxis general:

```
función nombre_función (nombre_de_parámetros_formales: tipo de dato) : tipo de dato resultado
constantes
variables   tipo de dato: nombre_de_variable1 (descripción de la variable1)
comienzo
      acción
      acción
      nombre_función = expresión
fin_función
```

Como ejemplo podemos indicar el del enunciado del ejercicio P2 de este examen.

P2. Dada la siguiente función recursiva

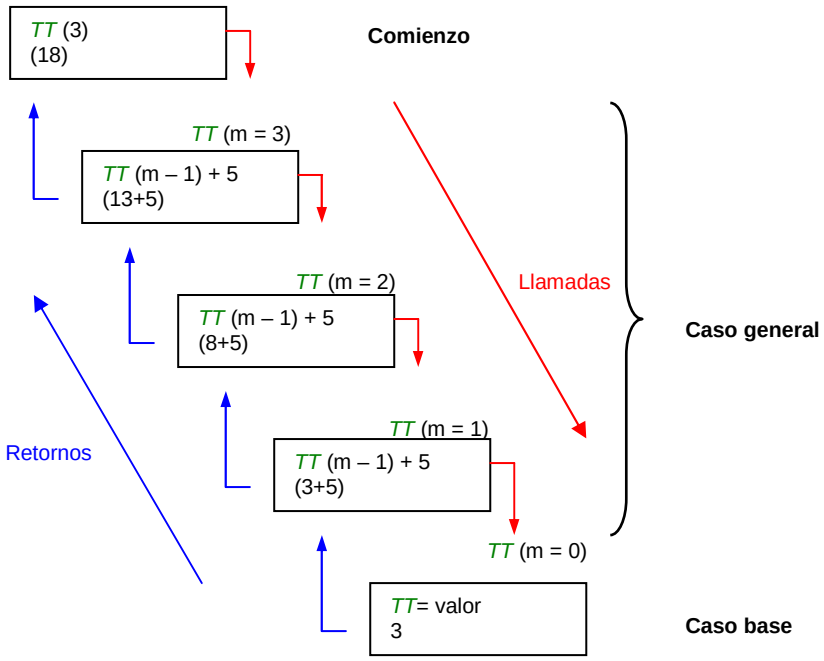
```
función  TT ( m: entero ): entero
variables entero: valor
comienzo
    sí m = 0 entonces
        hacer  valor ← 3
        sino
            hacer  valor ← TT ( m-1) + 5
    fin-si
    hacer TT ← valor
    escribir "Los valores de m y valor son " + m + "  |  "  + valor
fin-función
```

a) ¿Qué valor se producirá para m = 3?. **(0,5 punto)**
valor = 18

b) Indica las llamadas y retornos para m = 3. **(2 punto)**

La recursividad usa la pila. Una pila es una estructura de datos lineal en la cual sólo se accede a uno de sus extremos, llamado *cima*, tanto para insertar como para extraer datos.

	M		valor
4º	0	1º	entonces 3+ = 3
3º ↑	1	2º	entonces 3+5+ = 8
2º ↑	2	3º	entonces 3+5+5+ = 13
1º ↑	3	4º	entonces 3+5+5+5= 18

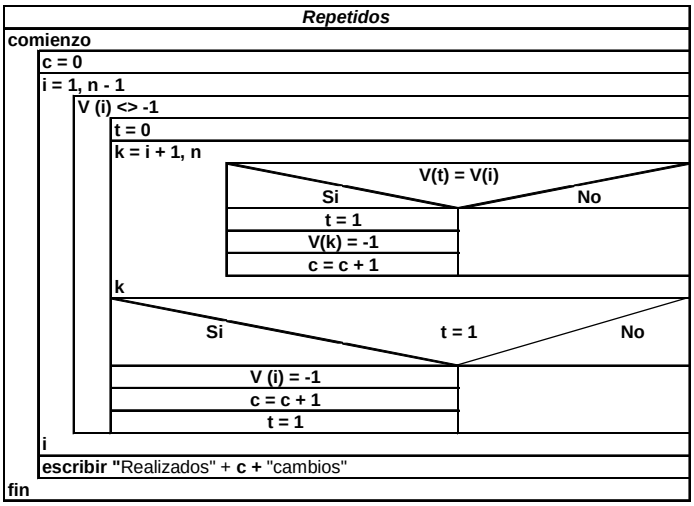


Comprobación realizada con Visual Basic:

Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

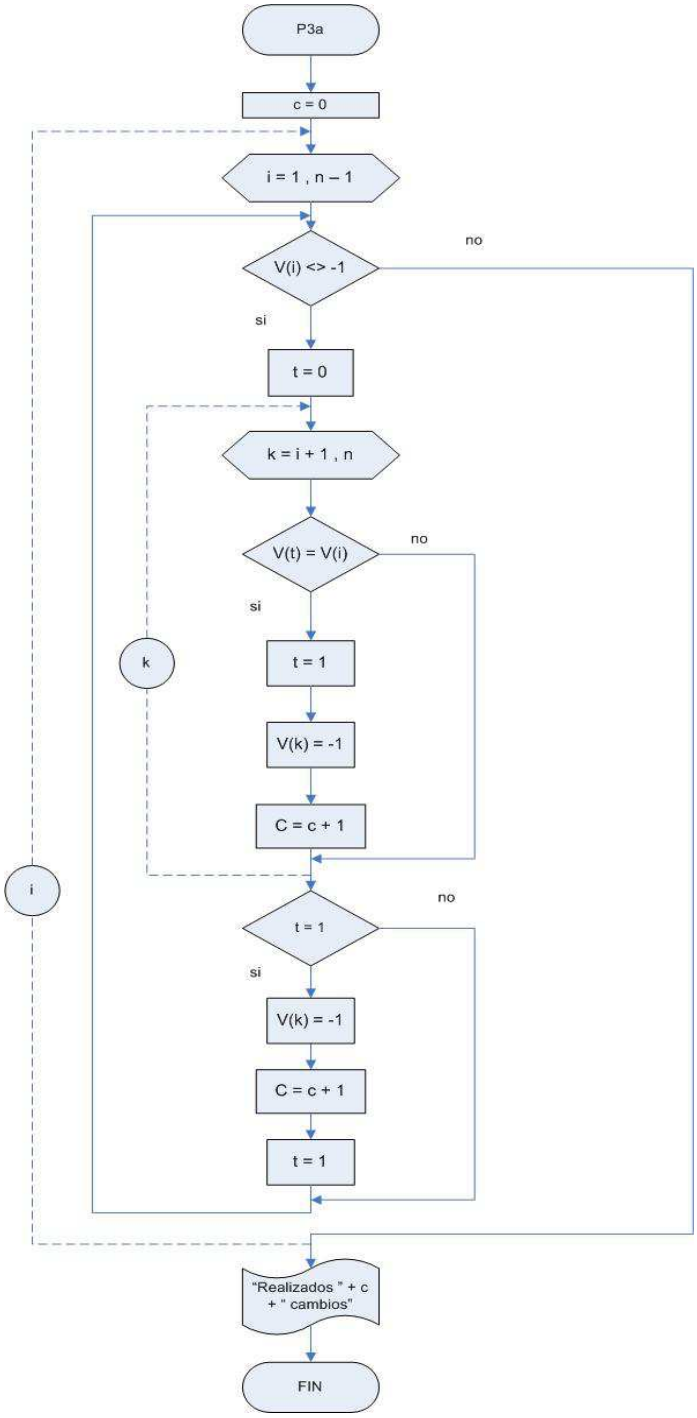
```
Option Explicit
Private Sub Form_Load()
    Dim x As Integer
    Dim y As Integer
    x = 3
    y = TT(x)
End Sub

Function TT(m As Integer) As Integer
Dim valor As Integer
    If m = 0 Then     'caso base
        valor = 3
    Else
        valor = TT(m - 1) + 5     'caso general o recursivo
    End If
    TT = valor
    MsgBox "Los valores de m y valor son: " & Format(m) & " | " & Format(valor)
End Function
```

P3. Dado el siguiente algoritmo en diagrama de Nassi-Shneiderman trasformarlo a pseudocódigo. En la declaración de variables no hace falta indicar el tipo ni la descripción. **(2 puntos)**



```
Algoritmo Título
variables
        tipo: c; i; n; V (i); t; k; V (t); V (k)
comienzo
        hacer c ← 0
        para i = 1 hasta n - 1
                mientras V(i) <> -1
                        hacer t ← 0
                        para k = i + 1 hasta n
                                si V(t) = V(i) entonces
                                        hacer t ← 1
                                        hacer V(k) ← -1
                                        hacer c← c + 1
                                fin-si
                        siguiente k
                        si t = 1 entonces
                                hacer V(i) ← -1
                                hacer c ← c + 1
                                hacer t ← 1
                        fin-si
                fin-mientras
        siguiente i
        escribir "Realizados " + c + " cambios."
fin
```

a) Transformar a diagrama de flujo. **(1 punto)**


P3a
c = 0
i = 1 , n – 1
V(i) <> -1
no
si
t = 0
k = i + 1 , n
V(t) = V(i)
no
si
k
t = 1
V(k) = -1
C = c + 1
i
t = 1
no
si
V(k) = -1
C = c + 1
t = 1
"Realizados " + c + " cambios"
FIN

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JULIO 2.009 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. Ejercicio teórico.

a) Dadas las funciones siguientes:

```
Función F1 (x : real): entero
Comienzo
    F1 ← x + 0.5
Fin-función.
```

```
Función F2 (x : entero): real
Comienzo
    F2 ← x  * 3.0
Fin-función.
```

Por medio de una traza indicar el valor de R al ejecutar la siguiente línea de un algoritmo: **(2 puntos)**
R ← **F2** (**F1** (**F1** (3.5))) 'Siendo R de tipo real

¡Ojo¡, la traza ha de hacerse como se ha explicado en clase o como las tenéis en la colección de ejercicios resueltos, no hay que inventar o innovar.

X	F1
3,5	4
4,0	4

X	F2
4	12,0

R
12,0

Comprobación realizada con Visual Basic:
En un formulario introducir una caja de texto (TextBox) de nombre txtResultado. Introducir un botón (CommandButton) de nombre cmdCalcular y copiar el código siguiente en la ventana 'Ver código' de Visual Basic.

```
Option Explicit
Dim R As Double

Private Sub cmdCalcular_Click()
R = F2(F1(F1(3.5)))
txtResultado.Text = R
End Sub

Function F1(x As Double) As Integer
   F1 = x + 0.5
End Function

Function F2(x As Integer) As Double
   F2 = x * 3
End Function
```

b) Escribir el resultado de la siguiente operación 16A + AE en hexadecimal, decimal y binario: **(1 punto)**

Si usamos por ejemplo el teorema fundamental de la numeración tenemos en el sistema decimal los siguientes valores para los términos de la suma:

16A→ 10*16^0 + 6*16^1 + 1*16^2 = 10 + 96 + 256 = 362_{10}
AE→ 14*16^0 + 10*16^1 = 14 + 160 = 174_{10}

si sumamos los dos términos

362_{10} + 174_{10} = 536_{10}

si pasamos a cada uno de los sistemas de numeración que se pide

Operación hexadecimal:

Divisiones	Cociente (div)	Resto (mod)
536/16	33	8
33/16	2	1
2/16	0	2

$16A + AE = 362_{10} + 174_{10} = 536_{10} = 218_{16}$

Operación decimal:

$16A + AE = 362_{10} + 174_{10} = 536_{10}$

Operación binario:

Divisiones	Cociente (div)	Resto (mod)
536/2	268	0
268/2	134	0
134/2	67	0
67/2	33	1
33/2	16	1
16/2	8	0
8/2	4	0
4/2	2	0
2/2	1	0
1/2	0	1

$16A + AE = 362_{10} + 174_{10} = 536_{10} = 1000011000_2$

c) ¿Cuales de las siguientes sentencias de asignación en pseudocódigo no son correctas? ¿Por qué?. **(1 punto)**

1) A + B ← a + b **No es correcta, signo + en el lado izquierdo.**

2) Cortante = Cortante + 1 **No es correcta, signo = (hay lenguajes que lo permiten, como el VB).**

3) 5 ← m **No es correcta, existe una constante en el lado izquierdo.**

4) Y + 5 ← 14 **No es correcta, signo + y constante en el lado izquierdo.**

d) Contesta a las siguientes preguntas.

1) ¿Qué es un compilador? **(0,5 puntos)**

Es un traductor que transforma cada instrucción del lenguaje de alto nivel a una o varias instrucciones del lenguaje máquina o de bajo nivel. Realizan la traducción de todo el programa fuente, generándose un programa objeto y una lista de errores, si los hay, al finalizar el intento de traducción. Si se modifica una instrucción del programa fuente hay que volver a compilar todo el programa.

2) ¿Para qué sirve la traza de un algoritmo? **(0,5 puntos)**

Es el procedimiento a seguir en las comprobaciones para saber que el algoritmo realiza lo que se supone que debe hacer. No es otra forma que ejecutar el algoritmo a mano, paso a paso, e ir anotando en una tabla los valores que van tomando las variables para cada caso. La traza es útil para encontrar errores en el diseño de los algoritmos, pero su validez es limitada, pues sólo se comprueban casos particulares, y sin embargo el algoritmo debe ser la solución general a una clase de problemas. Así mismo es una ayuda para demostrar la presencia de errores (si los encontramos) pero no su ausencia.

P2. Transformar la siguiente estructura de selección múltiple a:

1) Estructura equivalente de selección simple en pseudocódigo. **(1 punto)**
2) Diagrama de N-S del apartado 1). **(1 punto)**
3) Diagrama de flujo del apartado 1). **(1 punto)**

Caso expresión ***de***
Cte_1 **o (** Cte_2 **y** Cte_3 **)** ***entonces*** Sentencia_1
(Cte_4 **ó** Cte_5 **) y** Cte_6 ***entonces*** Sentencia_2
fincaso

Estructura equivalente de selección simple en pseudocódigo.

Teniendo en cuenta el orden de prioridad de los operadores lógicos y los paréntesis, tenemos:

- Primero lo que esta entre paréntesis.
- Segundo el operador “y”.
- Tercero el operador “o”.

```
comienzo
    si expresión = Cte_2 entonces
        si expresión = Cte_3 entonces
            hacer Sentencia_1
        fin-si
    fin-si
    si expresión = Cte_1 entonces
        hacer Sentencia_1
    sino
        si expresión = Cte_4 entonces
            si expresión = Cte_6 entonces
                hacer Sentencia_2
            fin-si
        fin-si
        si expresión = Cte_5 entonces
            si expresión = Cte_6 entonces
                hacer Sentencia_2
            fin-si
        fin-si
    fin-si
fin
```

Nota: existen más solución

Diagrama de N-S del apartado (1).

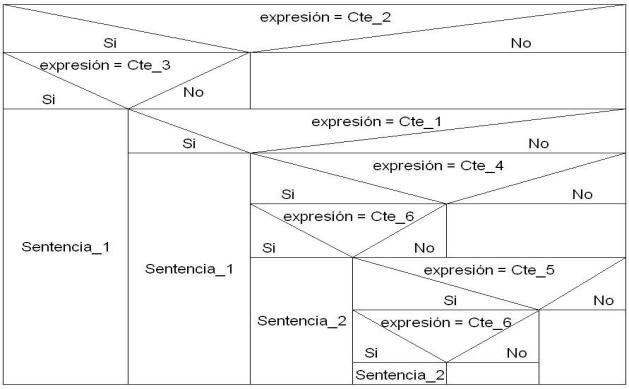
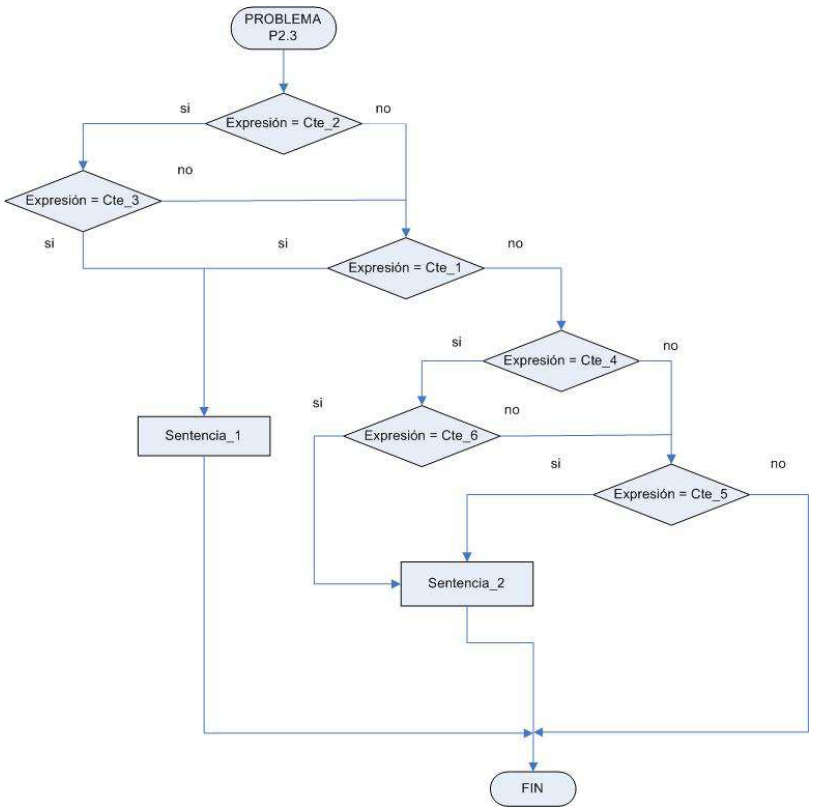


Diagrama de flujo del apartado (1).



P3. Realizar la traza del siguiente algoritmo, indicando los valores que toman todas las variables paso a paso como haría una máquina. **(1,5 puntos)**

```
Algoritmo P3
variables
        entero: a( 1 ....4); i;  j      //No se ha indicado la descripción porque no es necesaria en este ejercicio.
comienzo
        para i ← 1 hasta 4
              hacer a ( i ) ← 0
        siguiente i
        hacer a ( -1 ) ← 0
        hacer a ( 0 ) ← 0
        hacer a (1) ← 1
        para  i ← 1 hasta 4
                hacer a ( i ) ←1
              Para j ← 2 hasta  i - 1
                    hacer a ( j ) ← a ( j ) + a ( j - 1)
              siguiente j
              para j ← 1 hasta i
                    escribir a ( i )
                              escribir " "        //Espacio en blanco
              siguiente j
              escribir "                    "  //Salto de línea
         siguiente i
fin.
```

¡Ojo¡, la traza ha de hacerse como se ha explicado en clase o como las tenéis en la colección de ejercicios resueltos, no hay que inventar o innovar.

i	a(i)	a(-1)	a(0)	a(1)	i	a(i)	j	a(j)	j	escribe
$1 \rightarrow 4$	a(1) = 0									
2	a(2) = 0									
3	a(3) = 0									
4	a(4) = 0	0	0	1	$1 \rightarrow 4$	a(1) = 1	$2 \rightarrow 0$	a(2) = 0+1=1		
							1	a(1) = 1+0=1		
							0	a(0) = 0+0=0	$1 \rightarrow 1$	a(1)=**1** + espacio
										Línea en blanco
					2	a(2) = 1	$2 \rightarrow 1$	a(2) = 1 + 1 = 2		
							1	a(1) = 1+0=1	$1 \rightarrow 2$	a(2)=**2** + espacio

									2	a(2)=**2** + espacio	
										Línea en blanco	
					3	a(3) = 1	2 → 2	a(2) = 2+1=3	1 → 3	a(3)=**1** + espacio	
									2	a(3)=**1** + espacio	
									3	a(3)=**1** + espacio	
										Línea en blanco	
					4	a(4) = 1	2 → 3	a(2) = 3+1=4			Solución
							3	a(3) = 1+4=5	1 → 4	a(4)=**1** + espacio	**1**
									2	a(4)=**1** + espacio	**2 2**
									3	a(4)=**1** + espacio	**1 1 1**
									4	a(4)=**1** + espacio	**1 1 1 1**

a) La variable **a** del enunciado anterior, ¿representa una tabla bidimensional?, ¿por qué?. **(0,5 puntos)**

No, ya que una tabla bidimensional es un vector de vectores y éste no es el caso. Una tabla bidimensional representa una estructura compuesta por m filas y n columnas, que contienen un total de m*n elementos, cada uno de los cuales se identifica por dos índices entre paréntesis y separados por una coma. En este caso solamente tenemos un vector de nombre a de 4 elementos a(1...4) representado por a(i) y a(j). Los índices i y j representan la lectura de elementos del vector a(1...4).

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO MAYO 2.009 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. **a)** Dada la siguiente estructura de repetición

```
para x ← 1 hasta 3
    hacer A ← A + 1
    hacer B ← B - 1
siguiente x
```

Escribir las estructuras de repetición con condición inicial y final equivalente a la anterior. **(1 punto)**

Una solución sería:

```
hacer x ← 1
mientras x <= 3
  hacer A ← A + 1
  hacer B ← B - 1
  hacer x ← x + 1
fin-mientras
```

```
hacer x ← 1
repetir
    hacer A ← A + 1
    hacer B ← B – 1
    hacer x ← x + 1
hasta x > 3
```

para			mientras				repetir			
x	**A**	**B**	**x**	**A**	**B**	**x**	**x**	**A**	**B**	**x**
1 → 3	A + 1	B - 1	1	A + 1	B - 1	2	1	A + 1	B - 1	2
2	A + 2	B - 2		A + 2	B - 2	3		A + 2	B - 2	2
3	A + 3	B - 3		A + 3	B - 3	4		A + 3	B - 3	4

Otra solución sería:

```
hacer x ← 1
mientras x < 4
  hacer A ← A + 1
  hacer B ← B - 1
  hacer x ← x + 1
fin-mientras
```

```
hacer x ← 1
repetir
    hacer A ← A + 1
    hacer B ← B – 1
    hacer x ← x + 1
hasta x = 4
```

para			mientras				repetir			
x	**A**	**B**	**x**	**A**	**B**	**x**	**x**	**A**	**B**	**x**
1 → 3	A + 1	B - 1	1	A + 1	B - 1	2	1	A + 1	B - 1	2
2	A + 2	B - 2		A + 2	B - 2	3		A + 2	B - 2	2
3	A + 3	B - 3		A + 3	B - 3	4		A + 3	B - 3	4

b) ¿Cuántos dígitos son necesarios para representar los números usando un sistema de numeración de base n?, ¿qué ocurre si n > 10? **(1 punto)**

Son necesarios tantos dígitos como indique la base n. Ej.: base 2, dos dígitos, el 0 y el 1. base 8, ocho dígitos, el 0, 1, 2, 3, 4, 5, 6 y 7.
Si n > 10 entonces se usan como dígitos las letras A, B, C, etc así como tantos dígitos sean necesarios hasta completar la base n. Ej.: base 16(hexadecimal), dieciséis dígitos, el 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

c) Escribe en el sistema de numeración decimal los números 158_{11} y ABC_{16}. **(1 punto)**

Aplicando el teorema fundamental de la numeración:

$1 * 11^2 + 5 * 11^1 + 8 * 11^0 = 11^2 + 5 * 11 + 8 = 121 + 55 + 8 = 184$

$A * 16^2 + B * 16^1 + C * 16^0 = 10 * 16^2 + 11 * 16^1 + 12 * 16^0 = 2560 + 176 + 12 = 2748$

16	16	2560	$16^0 = 1$
x 16	x 11	+ 12	
196	16	2572	
+16	+ 16	+ 176	

256 176 2748

d) Contesta a las siguientes preguntas. **(1 punto)**

1) ¿Qué tarea tiene encomendada la unidad de control en la arquitectura de la máquina de John Von Neumann?

La de dirigir todas las operaciones del ordenador (las órdenes para el resto de los bloques) e interpretar las instrucciones recibidas; es el cerebro del ordenador.

2) ¿Qué es un byte?

Es la sucesión de dígitos binarios (Bits) adyacentes, tratados por el ordenador como una unidad de información. Es la información contenida en 8 Bits que equivale a un carácter, o llamado también octeto. Si combinamos 8 números binarios (Ej.: 01001010) hay exactamente 2^8 combinaciones, lo que da 256 posibilidades que es la cantidad de caracteres ASCII.

3) ¿Qué es la memoria RAM?

Es una memoria de lectura - escritura que a su vez es de acceso aleatorio. Es una memoria volátil, un corte en el fluido eléctrico hace desaparecer la información almacenada en ella.

4) ¿Qué es un subalgoritmo recursivo y de que partes consta la recursividad?.

Es aquel que tiene la característica de poderse llamar así mismo desde el interior del subalgoritmo. La potencia de un subalgoritmo recursivo se basa en la posibilidad de poder definir un número infinito de operaciones mediante un subalgoritmo recursivo finito. Consta de dos partes:

a. Una que utiliza el objeto que estamos definiendo (caso general o recursivo).

b. Y otra que no lo utiliza (caso base), siendo esta última la que nos permite abandonar la recursividad.

P2. Dado un número entero (númeroEntero), desarrollar un algoritmo en lenguaje natural que obtenga la suma de sus dígitos (sumaDígitos) y el número de dígitos (númeroDígitos). Como estructura de repetición únicamente se podrá usar la de condición final. Además no podrá usarse ninguna función interna de la sintaxis algorítmica ni los operadores aritméticos *mod, div* y ^. En su defecto, si así fuera, deberá diseñarse una función propia para calcular la parte entera (parteEntera). Tener en cuenta los filtros necesarios para que no se produzcan errores.

Ejemplo: númeroEntero = 362 sera igual a 2 + 6 + 3 = 11, sumaDígitos = 11 y númeroDígitos = 3.

Diseñar el algoritmo usando al menos las variables dadas entre paréntesis del enunciado. **(3 puntos)**

```
Algoritmo Suma de dígitos
variables entero: númeroEntero (número entero de entrada); númeroDígitos (contador de dígitos); sumaDígitos (suma de los dígitos)
          real: x (variable auxiliar de cálculo); x1 (variable auxiliar de cálculo); x2 (variable auxiliar de cálculo)
comienzo
    repetir
        aceptar númeroEntero
        si númeroEntero < 0 entonces
            mostrar "El número debe ser positivo."
            aceptar númeroEntero
        fin-si
    hasta númeroEntero >= 0
        hacer númeroDígitos ← 0
              sumaDígitos ← 0
        repetir
            hacer x ← númeroEntero / 10          // Si divido entre 10 y me quedo con la parte entera, así, sucesivamente, obtendré la
                                                    unidad, la decena, la centena ...
                  x1 ← parteEntera (x)  // Llamada a la función
                  x2 ← (x – x1) * 10
                  númeroEntero ← (númeroEntero – x2) / 10
                  sumaDígitos ← sumaDígitos + x2
                  númeroDígitos ← númeroDígitos + 1
        hasta x < 1// Hasta que la división sea menor a 1 con lo que ya sabré que he llegado al último dígito
        mostrar "La suma de dígitos es igual a " + sumaDígitos  + " y el número de dígitos es igual a " + númeroDígitos
fin

función parteEntera (y: real): entero   // Declaración de la función
variables entero: a (parte entera de y)
comienzo
        si y < 1 entonces
            hacer a ← 0
            sino
                hacer a ← 0
                repetir
                    hacer a ← a + 1
                hasta y < (a + 1)
        fin-si
        hacer parteEntera ← a
fin-función
```

a) Realizar la traza del apartado anterior para **númeroEntero** igual a 362, 13 y 0. **(1 puntos)**

númeroEntero	sumaDígitos	númeroDigitos	x	X1	X2	númeroEntero	sumaDígitos	númeroDígitos
362	0	0	36,2	36	2	36	2	1
			3,6	3	6	3	8	2
			0,3	0	3	0	**11**	**3**
13	0	0	1,3	1	3	1	3	1
			0,1	0	1	0	**4**	**2**
0	0	0	0	0	0	0	**0**	**1**

y	a	parteEntera
36,2	0	
	1	
	2	
	...	
	36	36
3,6	0	
	1	
	2	
	3	3
0,3	0	0
1,3	0	
	1	1
0,1	0	0
0	0	0

Comprobación realizada con Visual Basic:
En un formulario introducir dos cajas de texto (TextBox) una de nombre txtSuma y la otra de nombre txtDígitos. Introducir un botón (CommandButton) de nombre cmdCalcular y copiar el código siguiente en la ventana 'Ver código' del Visual Basic.

```
Dim númeroDígitos As Long   'Contador de los dígitos
Dim sumaDígitos As Long     'Suma de los dígitos
Dim númeroEntero As Long    'Número entero de entrada
'Dim otro As long           'Si queremos no usar una función para el cálculo de la parte entera
Dim x As Double, x1 As Double, x2 As Double 'Variables auxiliares de cálculo

Private Sub txtNumero_Change()
númeroEntero = Val(txtNumero.Text)
   If númeroEntero < 0 Then
      MsgBox "El número debe ser positivo."
      txtNumero.Text = ""
      txtSuma.Text = ""
      txtDígitos.Text = ""
   End If
End Sub

Private Sub cmdCalcular_Click()
   númeroDígitos = 0
   sumaDígitos = 0
   Do
      x = númeroEntero / 10   'Si divido entre 10 y me quedo con la parte entera, así, sucesivamente, obtendré la
unidad, la decena, la centena ...
      'otro = númeroEntero / 10   'Para no usar la función parteEntera
      x1 = parteEntera(x) 'Llamada a la función
      x2 = (x - x1) * 10
      númeroEntero = (númeroEntero - x2) / 10
      sumaDígitos = sumaDígitos + x2
      númeroDígitos = númeroDígitos + 1
   Loop Until x < 1    'Hasta que la división sea menor a 1 con lo que ya sabre que he llegado al último dígito
   txtSuma.Text = Str(sumaDígitos)
   txtDígitos.Text = Str(númeroDígitos)
End Sub

Function parteEntera(y As Double) As Long    'Declaración de la función
   Dim a As Long    'Parte entera de y
   If y < 1 Then
      a = 0
   Else
      a = 0
      Do
         a = a + 1
      Loop Until y < (a + 1)
   End If
   parteEntera = a
End Function
```

P3. Realizar la traza del siguiente algoritmo, indicando los valores que toman las variables i, j, a(i), a(j), a(1...4). **(1,5 puntos)**

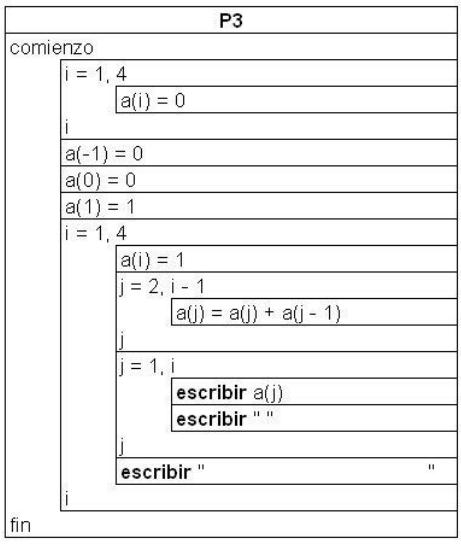
```
Algoritmo P3
variables
        entero: a( 1 ....4); i;  j      //No se ha indicado la descripción porque no es necesaria en este ejercicio.
comienzo
        para i ← 1 hasta 4
            hacer a ( i ) ← 0
        siguiente i
        hacer a ( -1 ) ← 0
        hacer a ( 0 ) ← 0
        hacer a (1) ← 1
        para  i ← 1 hasta 4
                hacer a ( i ) ←1
            Para j ← 2 hasta  i - 1
                hacer a ( j ) ← a ( j ) + a ( j - 1)
            siguiente j
            para j ← 1 hasta i
                escribir a ( j )
                        escribir " "        //Espacio en blanco
            siguiente j
            escribir "                      "  //Salto de línea
         siguiente i
fin.
```

i	a(i)	a(-1)	a(0)	a(1)	i	a(i)	j	a(j)	j	escribe
1 → 4	a(1) = 0									
2	a(2) = 0									
3	a(3) = 0									
4	a(4) = 0	0	0	1	1 → 4	a(1) = 1	2 → 0	a(2) = 0+1=1		
							1	a(1) = 1+0=1		
							0	a(0) = 0+0=0	1 → 1	a(1)=**1** + espacio
										Línea en blanco
					2	a(2) = 1	2 → 1	a(2) = 1 + 1 = 2		
							1	a(1) = 1+0=1	1 → 2	a(1)=**1** + espacio
									2	a(2)=**2** + espacio
										Línea en blanco
					3	a(3) = 1	2 → 2	a(2) = 2+1=3	1 → 3	a(1)=**1** + espacio
									2	a(2)=**3** + espacio
									3	a(3)=**1** + espacio
										Línea en blanco
					4	a(4) = 1	2 → 3	a(2) = 3+1=4		
							3	a(3) = 1+4=5	1 → 4	a(1)=**1** + espacio
									2	a(2)=**4** + espacio
									3	a(3)=**5** + espacio
									4	a(4)=**1** + espacio

Solución

1			
1	**2**		
1	**3**	**1**	
1	**4**	**5**	**1**

a) Transformar el algoritmo del apartado anterior en diagrama de Nassi/Schneiderman. **(0,5 puntos)**



Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO DICIEMBRE 2.008 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. **a)** Indicar cuales de las siguientes variables tienen nombre no válido y por qué. **(1 punto)**

1) AXIL-MAYORADO **no válido (-, no se puede usar)**
2) 4B **no válido (comienza por un número)**
3) combinación Pésima **no válido (hay un espacio)**
4) f_{cd} **no válido (hay subíndices)**

b) Si A = 1101_2 y B = 1011_2 que resultado se obtiene al analizar la siguiente expresión en base octal.
C = 2 * A + A – B **(2 punto)**

El 2, tanto en decimal como en octal, equivale en binario a 10, por lo tanto podemos optar por trabajar sumando o multiplicando para resolver la operación:

2 * A)

```
  1101
 x  10
------
  0000
+ 1101
------
 11010
```

2 * A + A)

```
 11010
+ 1101
------
100111
```

2 * A + A - B)

```
 100111
- 1011
------
 011100
```

Si 0 * 0 = 0, 0 * 1=0, 1 * 0 = 0, 1 * 1 = 1; 0 + 0 =0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 0 y arrastro 1 y 0 - 0 = 0, 0 - 1 = no cabe o se pide prestado al próximo 1 y me llevo 1, 1 - 0 = 1 y 1 - 1 = 0

entonces, pasándolo a octal (grupos de tres cifras desde la derecha añadiendo ceros hasta completar un grupo de tres con el último dígito de la izquierda) tenemos 011 / 100, que equivale a 34_8

c) Contesta a las siguientes preguntas. **(1 punto)**

1) ¿Qué es un byte?
Es el número de bits necesarios para almacenar un carácter, por lo general un byte es sinónimo de 8 bits u octeto.

2) ¿Qué es un compilador?
Es un traductor que transforma cada instrucción del lenguaje de alto nivel en instrucciones de lenguaje máquina.

3) Las variables tipo arrays se identifican por medio de ¿qué?
Los arrays se identifican por medio de su nombre, sus elementos, su índice, su dimensión, su longitud y su tipo.

4) Los índices i y j pueden ser un subrango (v_i...v_f) de valores de tipo ordinal. V_i es el límite inferior y v_f el límite superior dentro de los valores que toma el índice. ¿Qué nos indica la expresión ($v_{f1} – v_{i1} + 1$) * ($v_{f2} – v_{i2} + 1$)?
El número total de elementos de la matriz.

P2. Definir un procedimiento que obtenga la división entera y el resto de la misma donde el dividendo y el divisor pueden ser datos de tipo real, utilizando para ello únicamente los operadores aritméticos *suma* y *resta* y la estructura de repetición *repetir*. No podrá usarse ninguna función interna de la sintaxis algorítmica. Tener en cuenta los filtros necesarios para que no se produzcan errores. **(3 puntos)**

La división se puede considerar como una sucesión de restas. El algoritmo trata de contar cuántas veces se puede restar el divisor al dividendo y dicho contador sería el cociente. Cuando ya no se pueda restar más sin que salga un número positivo, se tendrá el resto.

```
procedimiento DivisiónEntera (dividendo, divisor (E): real; cociente, resto (S): entero)
variables entero: x (dividendo entero); y (divisor entero)
comienzo
        hacer x ← 1
              y ← 1
        repetir
            hacer x ← x + 1
        hasta dividendo < x + 1 // Nos quedamos con la parte entera del dividendo
        repetir
            hacer y ← y + 1
        hasta divisor < y + 1      // Nos quedamos con la parte entera del divisor
        hacer dividendo ← x
        hacer divisor ← y
        hacer cociente ← 0
        repetir
            hacer dividendo ← dividendo - divisor
                  cociente ← cociente + 1
        hasta dividendo < divisor
        hacer resto ← dividendo
fin procedimiento
```

a) Realizar la traza del apartado anterior para dividendo = 10,53 y divisor = 3,3. **(1 puntos)**

dividendo	divisor	x	y	dividendo	divisor	cociente	dividendo	cociente	resto
10,53	3,3	1	1	10	3	0	7	1	
		2	2				4	2	
		3	3				1	**3**	**1**
		4							
		5							
		6							
		7							
		8							
		9							
		10							

Comprobación realizada con Visual Basic:
Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic. El separador de decimales es la coma.

```
Option Explicit

Private Sub Form_Load()
   Dim dividendo As Single 'Es el dividendo de la división
   Dim divisor As Single   'Es el divisor de la división
   Dim cociente As Integer 'Es el cociente entero de la división
   Dim resto As Integer    'Es el resto de la división
   Dim repetir As Integer  'Interruptor

comienzo:
   dividendo = Val(InputBox("Dividendo= "))
   While dividendo <= 0
      If dividendo <= 0 Then
         MsgBox "El dividendo debe ser mayor que cero."
         dividendo = Val(InputBox("Dividendo= "))
      End If
   Wend
   divisor = Val(InputBox("Divisor= "))
   While divisor <= 0
      If divisor <= 0 Then
         MsgBox "El divisor debe ser mayor que cero."
         divisor = Val(InputBox("Divisor= "))
      End If
   Wend
   Call DivisiónEntera(dividendo, divisor, cociente, resto)    'Llamada al procedimiento
   MsgBox ("Cociente= " & cociente & vbCrLf & "Resto= " & resto)
   repetir = MsgBox("¿Quiero volver a calcular el cociente y el resto entero de la división? ", vbQuestion + vbYesNo +
vbDefaultButton2, "Problema 2")
```

```
   If repetir = 6 Then    '6 valor devuelto por el parámetro vbYesNo al pulsar sobre el botón 'Si'
      GoTo comienzo
   End If
End Sub

'Aquí esta la declaración del procedimiento, la parte que se pide en el enunciado del problema 2.

Sub DivisiónEntera(dividendo, divisor, cociente, resto)
   Dim x As Integer    'Es el dividendo entero de la división
   Dim y As Integer    'Es el divisor entero de la división
   x = 1
   y = 1
   Do
      x = x + 1
   Loop Until dividendo < x + 1
   Do
      y = y + 1
   Loop Until divisor < y + 1
   dividendo = x
   divisor = y
   cociente = 0
   Do
      dividendo = dividendo - divisor
      cociente = cociente + 1
   Loop Until dividendo < divisor
   resto = dividendo
End Sub
```

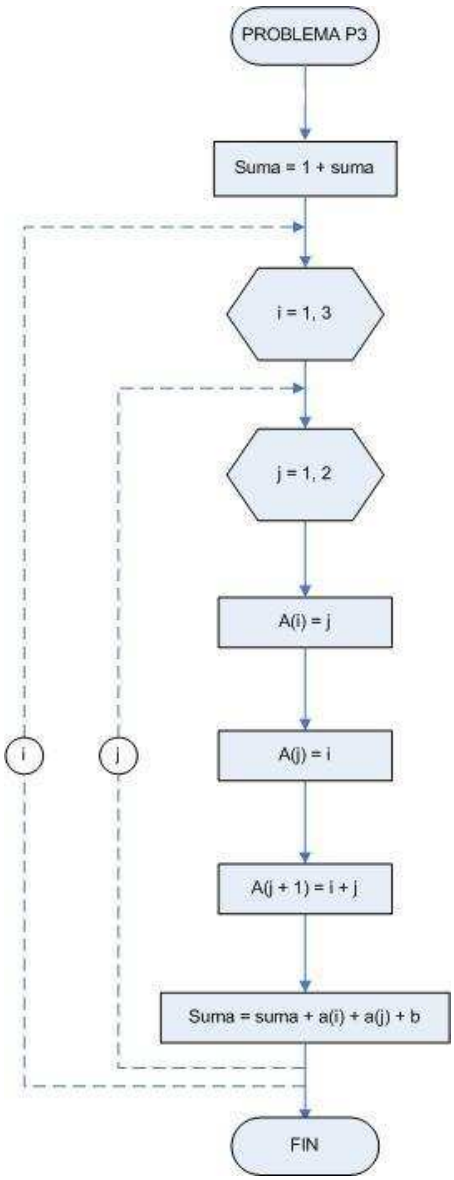
P3. Realizar la traza del siguiente algoritmo, indicando los valores que toman las variables i, j, a(i), a(j), a(j +1), b y suma. **(1 puntos)**

```
Algoritmo Traza
variables      entero: a(i), a(j), a(j +1); i , j; suma; b      // No se ha indicado la descripción por no ser necesaria
comienzo
        hacer suma ← 1 + suma
        para i ← 1 hasta 3
              para j ← 1 hasta 2
                    hacer     a(i) ← j
                              a(j) ← i
                              a(j +1) ← i + j
                              suma ← suma + a(i) + a(j) + b
              siguiente j
        siguiente i
fin
```

suma	i	j	a(i)	a(j)	a(j +1)	suma
1	1 → 3	1 → 2	a(1) = 1	a(1) = 1	a(2) = 2	3
		2	a(1) = 2	a(2) = 1	a(3) = 3	6
	2	1 → 2	a(2) = 1	a(1) = 2	a(2) = 3	11
		2	a(2) = 2	a(2) = 2	a(3) = 4	15
	3	1 → 2	a(3) = 1	a(1) = 3	a(2) = 4	19
		2	a(3) = 2	a(2) = 3	a(3) = 5	27

Una vez que se declara una variable, el sistema le asigna un valor por defecto, ya que la variable esta creada y además con su tipo de dato. Así, que si se usa la variable 'suma' o 'b', ésta contiene en memoria un dato, que esta claro que si uno no lo inicializa al valor que le interesa contendrá el de defecto, en este caso suma = 0 y b = 0. No es normal usar el valor por defecto, ya que casi siempre se necesita si la variable es de tipo entero que sea distinto a cero.

a) Transformar el algoritmo del apartado anterior en diagrama de flujo. **(1 puntos)**


PROBLEMA P3
Suma = 1 + suma
i = 1, 3
j = 1, 2
A(i) = j
A(j) = i
A(j + 1) = i + j
Suma = suma + a(i) + a(j) + b
FIN
i
j

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO SEPTIEMBRE 2.008 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. **a)** Deducir el resultado de las expresiones paso a paso como operaría un ordenador. **(1,5 puntos)**

1) – 4 * 7 + 2 ^3 / 4 – 5

- 4 * 7 + 2 ^3 / 4 -5
- 4 * 7 + 8 /4 –5
- 28 + 8 / 4 - 5
- 28 + 2 - 5
- 31

2) 7 * 10 –15 mod 3 *4 + 9

7 * 10 –15 mod 3 * 4 + 9
70 –15 mod 3 * 4 + 9
70 – 0 * 4 + 9
70 - 0 + 9
79

3) (C <= D + 7) o (7 > 5) para cualquier valor de C y D

devuelve verdadero

4) (4,5 > x) y (z < x + 7,5) para x = 7 y z = 5

4,5 > 7 es falso
5 < 7 + 7,5 es verdadero
(falso) y (verdadero)
falso

5) no (z > 14) para z = 7

verdadero

6) (redondeo (5,5) < x) o (x <= 5,5) para x = 5

redondeo (5,5) < 5 es falso
5 <= 5,5 es verdadero
(falso) o (verdadero)
verdadero

b) Realizar la conversión del número $11011111,11111_2$ a base octal. **(1 punto)**

	Agrupación	Equivalente octal
Resultado: $337,76_8$ Observar como ha sido necesario añadir un cero en la última agrupación de la parte entera y otro en la parte fraccionaria para completar los grupos de 3 dígitos.	011	3
	011	3
	111	7
	,	,
	111	7
	110	6

c) De qué partes se compone la arquitectura de la máquina de John Von Neumann y explicar el funcionamiento de cada parte. **(1 punto)**

La arquitectura de la máquina de John Von Neumann consta de las siguientes partes:

i. **Unidad de control (UD): Constituye el ¨cerebro¨ del ordenador. Se encarga de dirigir todas las operaciones del mismo (las órdenes para el resto de los bloques) e interpretar las instrucciones recibidas.**
ii. **Unidad aritmético lógica(ALU): Ejecuta operaciones aritméticas y lógicas según los datos o instrucciones recibidas de los programas. Suma, resta, multiplica, divide, niega sentencias, realiza comparaciones, etc.**

iii. Memoria principal o central (CM): Almacena la información . Contiene los datos y programas que van a ser ejecutados.

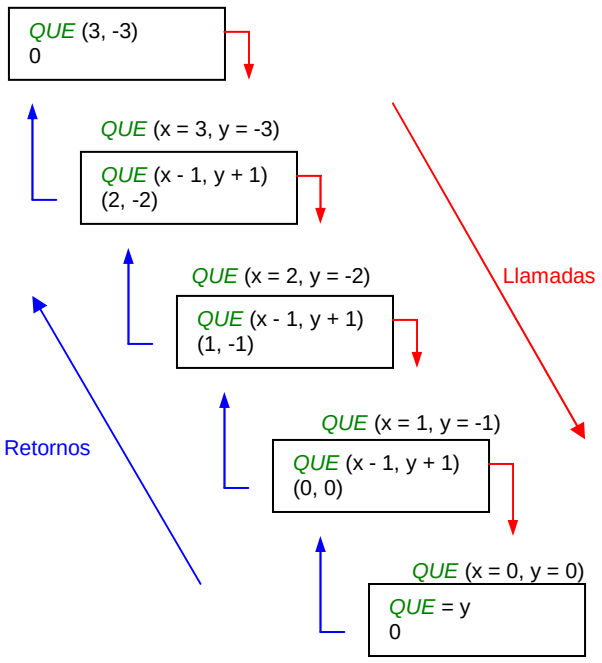
d) Realizar la traza del siguiente algoritmo, indicando los valores que toman las variables i, j, a(i), a(j), a(j +1) y suma. **(1,5 puntos)**

```
Algoritmo Traza
variables      entero: a(i), a(j), a(j +1); i , j; suma        // No se ha indicado la descripción por no ser necesaria
comienzo
    hacer suma ← 1
    para i ← 1 hasta 3
        para j ← 1 hasta 2
            hacer    a(i) ← j
                     a(j) ← i
                     a(j +1) ← i + j
        siguiente j
    siguiente i
fin
```

suma	i	j	a(i)	a(j)	a(j +1)
1	1 → 3	1 → 2	a(1) = 1	a(1) = 1	a(2) = 2
		2	a(1) = 2	a(2) = 1	a(3) = 3
	2	1 → 2	a(2) = 1	a(1) = 2	a(2) = 3
		2	a(2) = 2	a(2) = 2	a(3) = 4
	3	1 → 2	a(3) = 1	a(1) = 3	a(2) = 4
		2	a(3) = 2	a(2) = 3	a(3) = 5

P2. Dada la siguiente función recursiva, realizar el gráfico de llamadas y retornos para el caso de QUE (3, -3). **(1,5 puntos)**

```
función QUE (x, y: entero): entero
comienzo
    si x = 0 entonces
        si y > x entonces
            hacer QUE ← QUE (y – 1, x) + 1
            sino
                hacer QUE ← y
        fin si
        sino
            hacer QUE ← QUE (x – 1, y + 1)
    fin si
fin función
```



Comprobación realizada con Visual Basic:
Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Private Sub Form_Load()
```

```
    Dim x As Integer
    Dim y As Integer
    Dim z As Integer
    x = 3
    y = -3
    z = QUE(x, y)
    MsgBox "Resultado de la función: " & Format(z)
End Sub

Function QUE(x1 As Integer, y1 As Integer) As Integer
    If x1 = 0 Then
        If y1 > x1 Then
            QUE = QUE(y1 - 1, x1) + 1
        Else
            QUE = y1
        End If
    Else
            QUE = QUE(x1 - 1, y1 + 1)
    End If
End Function
```

P3. Desarrollar un algoritmo en pseudocódigo que permita, dado una lista **V** de **n** elementos positivos, que pueda tener algunos de ellos repetidos, reemplazar cada elemento repetido por -1 e indique el número de modificaciones realizadas. La única estructura de repetición que puede usarse es el bloque de control **mientras**. Tener en cuenta los filtros necesarios para que no se produzcan errores. Ejemplo:

V = (2, 4, 2, 4, 0); quedando tras la lectura como (2, 4, -1, -1, 0); número de modificaciones **c** = 2

Diseñar el algoritmo usando al menos las variables dadas en el enunciado. **(2,5 puntos)**

```
Algoritmo Problema P3
variables entero: n (nº de elementos del vector V); k (contador auxiliar de elementos); c (contador de cambios realizados);
      i (contador de elementos de V), V(i) (vector de i elementos); V(k) (vector auxiliar de comparación)
comienzo
      mostrar "Introduzca el número de elementos de la lista V: "
      aceptar n
      mientras n <= 0
            mostrar "El número de elementos debe ser positivo y mayor que 0, introduzca de nuevo el número de elementos:"
            aceptar n
      fin mientras
      hacer i ← 1
      mientras i <= n
            mostrar "Introduzca el valor del elemento V(" + i + "):"
            aceptar V(i)
            mientras V(i) < 0
                  mostrar "El valor del elemento debe ser positivo, vuelva a introducir el valor del elemento V(" & i & "):"
                  aceptar V(i)
            fin mientras
            hacer i ← i + 1
      fin mientras
      hacer c ← 0
      hacer i ← 2
      mientras i <= n
            si V(i) <> -1 entonces
                  hacer k ← i - 1
                  mientras k >= 1
                        si V(i) = V(k) entonces
                              hacer V(i) ← - 1
                              hacer c ← c + 1
                        fin si
                        hacer k ← k – 1
                  fin mientras
            fin si
            hacer i ← i + 1
      fin mientras
      mostrar "El número de cambios realizados es de " + c
      hacer i ← 1
      mientras i <= n
            mostrar "Estado de la lista final con los cambios, elemento V(" + i + ")= " + V(i)
            hacer i ← i + 1
      fin mientras
fin
```

a) Realizar la traza para n = 5 y V = (0, 1, 1, 3, 1). **(1 puntos)**

n	i	V(i)	c	k	V(i)

5	1	0			
	2	1			
	3	1			**-1**
	4	3			
	5	1			**-1**
	6				
	2		0	1	
	3			0	
	4		1	2	
	5			1	
	6			0	
				3	
				2	
				1	
				0	
				4	
			2	3	
				2	
				1	
				0	

Comprobación realizada con Visual Basic:

Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Dim n As Integer    'Número de elementos del vector V
Dim k As Integer    'Contador auxiliar de elementos
Dim c As Integer    'Contador de cambios realizados
Dim i As Integer    'Contador de elementos de V
Dim V() As Integer  'Vector de n elementos

Private Sub Form_Load()
   n = Val(InputBox("Introduzca el número de elementos de la lista V:"))
   While n <= 0
      n = Val(InputBox("El número de elementos debe ser positivo y mayor que 0, introduzca de nuevo el número de elementos:"))
   Wend
   ReDim V(n)   'Reasigna espacio de almacenamiento para variables de matriz dinámica
   i = 1
   While i <= n
      V(i) = Val(InputBox("Introduzca el valor del elemento V(" & i & "):"))
      While V(i) < 0
         V(i) = Val(InputBox("El valor del elemento debe ser positivo, vuelva a introducir el valor del elemento V(" & i & "):"))
      Wend
      i = i + 1
   Wend
   c = 0
   i = 2
   While i <= n
      If V(i) <> -1 Then
         k = i - 1
         While k >= 1
            If V(i) = V(k) Then
               V(i) = -1
               c = c + 1
            End If
            k = k - 1
         Wend
      End If
      i = i + 1
   Wend
   MsgBox "El número de cambios realizados es de " & c
   i = 1
   While i <= n
      MsgBox "Estado de la lista final con los cambios, elemento V(" & i & ")=" & V(i)
      i = i + 1
   Wend
End Sub
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JUNIO 2.008 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. **a)** ¿Cuales de las siguientes sentencias de asignación no son correctas? ¿Por qué?. **(1,5 puntos)**

1) A + B ← a + b **No es correcta, signo + en el lado izquierdo.**
2) Cortante = Cortante + 1 **No es correcta, signo = (hay lenguajes que lo permiten, como el VB).**
3) Mf ← 45000
4) K ← K – 4
5) X ← 18 – X
6) 5 ← m **No es correcta, existe una constante en el lado izquierdo.**
7) Suma ← caudal1 + caudal2
8) Y + 5 ← 14 **No es correcta, signo + y constante en el lado izquierdo.**

d) En una clase hay 100 alumnos, de los cuales 24 son chicos y 32 chicas. ¿En qué base numérica la frase anterior es cierta y por qué? **(1 punto)**

$(2*b^1 + 4*b^0) + (3*b^1 + 2*b^0) = 1*b^2 + 0*b^1 + 0*b^0$;
$4 + 2*b + 2 + 3*b = b^2$;
$b^2 – 5*b – 6 = 0$;
b = 6 y –1 por lo que la solución correcta será b = 6.

d) Indicar al menos 3 diferencias entre procedimientos y funciones. **(1,5 puntos)**

- **Mientras que a un procedimiento se le llama mediante una instrucción de llamada a procedimiento (no es necesario, se puede usar su nombre para realizar la llamada), a una función se la llama usando su nombre en una expresión.**

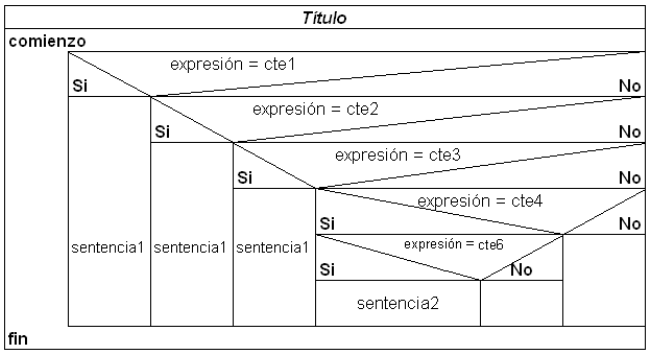
- **Puesto que se debe asociar un valor al nombre de una función, también se le debe asociar un tipo, por tanto, la cabecera de una función debe incluir un identificador que identifique el tipo del resultado. Sin embargo, no se asocia ningún valor con el nombre de un procedimiento y, por tanto, tampoco ningún tipo.**

- **Las funciones normalmente devuelven un único valor al algoritmo principal que la llama. Los procedimientos suelen devolver más de un valor, o pueden no devolver ninguno si solamente realizan alguna tarea, como una operación de salida.**

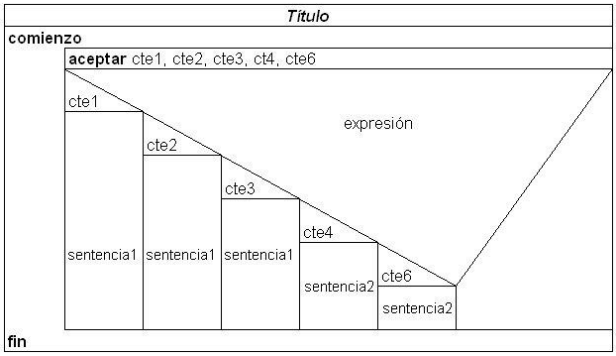
- **En los procedimientos, los valores se devuelven a través de parámetros por variable, pero el valor de una función se devuelve mediante la asignación al nombre de la función de dicho valor en la parte de instrucciones de la definición de la función.**

P2. **Transformar la siguiente estructura de selección múltiple en diagrama de Nassi Schneiderman. (1,5 puntos)**

```
comienzo
    caso expresión de
        cte1 ó cte2 ó cte3 entonces sentencia1
        cte4 y cte6 entonces sentencia2
    fin caso
fin
```



Forma 1



Forma 2

P3. Desarrollar un algoritmo en pseudocódigo que permita, dada una tabla **T** de **f** filas y **c** columnas, introducir números enteros y decimales de elementos positivos y negativos. Que localice y cuente el número de elementos positivos y por medio de una función de nombre **Suma** sume el valor de cada uno de los elementos positivos reales previamente redondeados al entero superior. Tener en cuenta los filtros necesarios para que no se produzcan errores. Ejemplo:

$$T = \begin{pmatrix} -2{,}45 & 0{,}99 \\ 0 & 1{,}01 \\ 3 & -2 \end{pmatrix}$$

Npositivos: 4, **Spositivos**: ((0,99) entero superior = 1) + ((1,01) entero superior = 2) = 3

Diseñar el algoritmo usando al menos las variables dadas en el enunciado. **(2,5 puntos)**

```
Algoritmo Problema P3
Variables entero: f (nº de filas); c (nº de columnas); i , j (contadores); Npositivos (nº de elementos positivos de la tabla),
                  Spositivos (suma de elementos positivos y reales previamente redondeados al entero superior); Nredondeado
                  (nº real redondeado al entero superior); acumulador (acumula valores)
          real: T ( i , j ) (matriz de dimensión f , c)
comienzo
    mostrar "Introduzca el número de filas de la tabla: "
    aceptar f
    mostrar "Introduzca el número de columnas de la tabla: "
    aceptar c
        mientras f <= 0 o c <= 0
            si f <= 0 entonces
                    mostrar "El número de filas debe ser superior a 0, introduzca de nuevo el valor para f:"
                    aceptar f
                    sino
                        mostrar "El número de columnas debe ser superior a 0, introduzca de nuevo el valor para f:"
                        aceptar c
            fin si
        fin mientras
        hacer Npositivos ← 0
        hacer Spositivos ← 0
        hacer acumulador ← 0
        para i ← 1 hasta f
            para j ← 1 hasta c
                aceptar T ( i , j )
                si T ( i , j ) >= 0 entonces
                    hacer Npositivos ← Npositivos + 1
                    si Ent(T ( i , j )) <> T ( i , j ) entonces
                        hacer  Nredondeado  ← Ent(T ( i , j ) + 1)
                        hacer Spositivos ← Suma (Nredondeado)
                    fin si
                fin si
            siguiente j
        siguiente i
        mostrar "Número de elementos positivos de la tabla: " + Npositivos + "; " + "suma del valor de los elementos positivos
                reales previamente redondeados al entero superior: " + Spositivos
fin

 ----------Declaración de la función----------
 Función Suma (parámetro: entero): entero
 comienzo
     hacer  acumulador  ← acumulador + parámetro
     hacer  Suma  ← acumulador
 fin-función
```

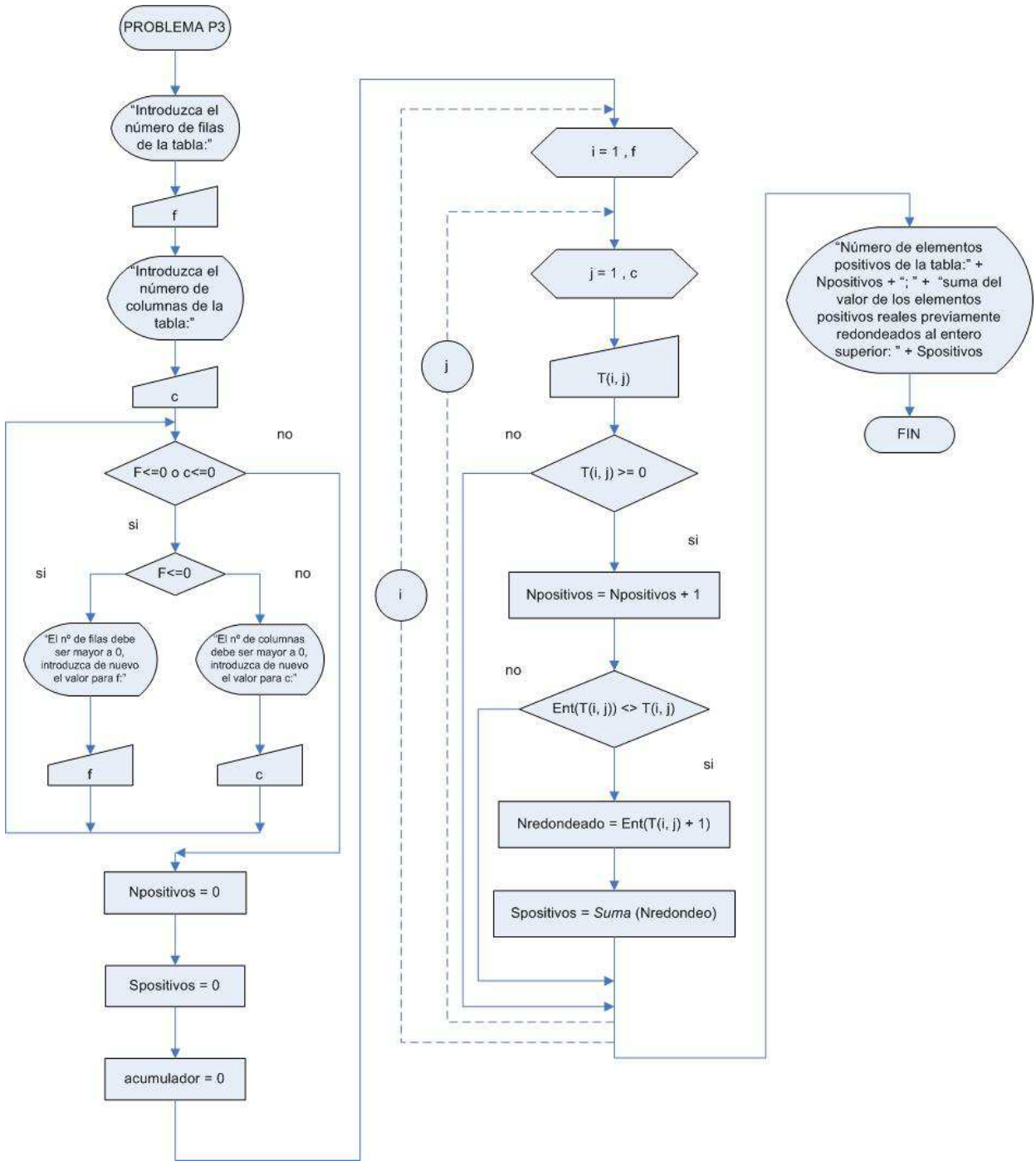
a) Realizar la traza para f= 2, c= 2 y $T = \begin{pmatrix} -2{,}01 & 0{,}99 \\ 0{,}01 & 10 \end{pmatrix}$. **(1 puntos)**

ALGORITMO PRINCIPAL											SUBALGORITMO	
f	**c**	**Npositivos**	**Spositivos**	**acumulador**	**i**	**j**	**T(i, j)**	**Npositivos**	**Nredondeado**	**Spositivos**	**parámetro**	**acumulador**
2	2	0	0		1→2	1→2	-2,01					
						2	0,99	1	1	1	1	1
					2	1→2	0,01	2	1	**2**	1	2
						2	10	**3**				

Comprobación realizada con Visual Basic:
Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Dim m As Integer    'Dimensión de las filas de la matriz A
Dim f As Integer   'Número de filas de la tabla T
Dim c As Integer 'Número de columnas de la tabla T
Dim i As Integer   'Contador de filas
Dim j As Integer   'Contador de columnas
Dim T() As Double  'Matriz de dimensiones f x c
Dim Npositivos As Integer   'Número de elementos positivos de la tabla
Dim Spositivos As Integer   'Suma de elementos positivos reales previamente redondeados al entero superior de la tabla
Dim Nredondeado As Integer  'Número real redondeado al entero superior
Dim acumulador As Integer   'Variable auxiliar para acumular valores enteros

Private Sub Form_Load()
   f = Val(InputBox("Introduzca el número de filas de la tabla T:"))
   c = Val(InputBox("Introduzca el número de columnas de la tabla T:"))
   While f <= 0 Or c <= 0
      If f <= 0 Then
         f = Val(InputBox("El número de filas debe ser superior a 0, introduzca de nuevo el valor para f:"))
         Else
         c = Val(InputBox("El número de columnas debe ser superior a 0, introduzca de nuevo el valor para c:"))
      End If
   Wend
   ReDim T(f, c)   'Reasigna espacio de almacenamiento para variables de matriz dinámica
   Npositivos = 0
   Spositivos = 0
   acumulador = 0
   For i = 1 To f
      For j = 1 To c
         T(i, j) = Val(InputBox("Elemento (" & i & ", " & j & ")="))
         If T(i, j) >= 0 Then
            Npositivos = Npositivos + 1
            If Int(T(i, j)) <> T(i, j) Then
               Nredondeado = Int(T(i, j) + 1)
               Spositivos = Suma(Nredondeado)
            End If
         End If
      Next j
   Next i
   MsgBox "Número de elementos positivos de la tabla: " & Npositivos & vbCrLf & "Suma del valor de los elementos reales positivos previamente redondeados al entero superior: " & Spositivos
End Sub

Function Suma(parámetro As Integer) As Integer
   acumulador = acumulador + parámetro
   Suma = acumulador
End Function
```

b) Transformar el algoritmo desarrollado en diagrama de flujo. **(1 puntos)**


PROBLEMA P3
"Introduzca el número de filas de la tabla:"
f
"Introduzca el número de columnas de la tabla:"
c
F<=0 o c<=0
no
si
F<=0
si
no
"El nº de filas debe ser mayor a 0, introduzca de nuevo el valor para f:"
"El nº de columnas debe ser mayor a 0, introduzca de nuevo el valor para c:"
f
c
Npositivos = 0
Spositivos = 0
acumulador = 0
i
i = 1 , f
j
j = 1 , c
T(i, j)
no
T(i, j) >= 0
si
Npositivos = Npositivos + 1
no
Ent(T(i, j)) <> T(i, j)
si
Nredondeado = Ent(T(i, j) + 1)
Spositivos = Suma (Nredondeo)
"Número de elementos positivos de la tabla:" + Npositivos + "; " + "suma del valor de los elementos positivos reales previamente redondeados al entero superior: " + Spositivos
FIN

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO DICIEMBRE 2.007 (Duración del examen: 2:30h)

APELLIDOS:	Nº DE LA CONVOCATORIA A LA QUE SE PRESENTA:
NOMBRE:	

P1. **a)** Dada la siguiente expresión aritmética en base binaria:

(1 1 0 1 1 0 1 0 1) x (1 0 1 1) = (1 0 0 0 + 1 0 + 1) x (1 1 0 1 1 0 1 0 1) = 1 0 0 0 x 1 1 0 1 1 0 1 0 1 + 1 0 x 1 1 0 1 1 0 1 0 1 + 1 x 1 1 0 1 1 0 1 0 1 = 1 0 0 1 0 1 1 0 0 0 1 1 1

¿Es correcta la operación?, ¿por qué?. **(1,50 puntos)**

Sí. Ya que,

```
  1 1 0 1 1 0 1 0 1
    x     1 0 1 1
  -----------------
  1 1 0 1 1 0 1 0 1
1 1 0 1 1 0 1 0 1
0 0 0 0 0 0 0 0 0
1 1 0 1 1 0 1 0 1
-----------------
1 0 0 1 0 1 1 0 0 0 1 1 1
```

La multiplicación es inmediata; nunca se lleva algo como en la decimal. Así; 1 x 1 = 1, 1 x 0 = 0, 1 x 1 = 1, 1 x 0 = 0,1 x 1 = 1,1 x 1 = 1,1 x 0 = 0,1 x 1 = 1 y 1 x 1 = 1, por lo que la multiplicación con el primer dígito nos queda 1 1 0 1 1 0 1 0 1. Para el siguiente dígito se multiplica corriendo un lugar hacia la izquierda, como en los decimales. Para el tercer dígito, como es cero, sus productos son siempre ceros. Para el cuarto dígito, se multiplica considerando siempre el respectivo corrimiento a la izquierda. Finalmente, se realiza la suma de la manera ya conocida.

Un método mas rápido y que usualmente se utiliza en ordenadores es notando que un número binario, 1 0 1 1, representa la suma 1 0 0 0 + 1 0 + 1 y que al multiplicarlo por otro binario, el resultado es la suma de las multiplicaciones de cada elemento por el número multiplicado, esto es:

(1 1 0 1 1 0 1 0 1) x (1 0 1 1) = (1 0 0 0 + 1 0 + 1) x (1 1 0 1 1 0 1 0 1) = 1 0 0 0 x 1 1 0 1 1 0 1 0 1 + 1 0 x 1 1 0 1 1 0 1 0 1 + 1 x 1 1 0 1 1 0 1 0 1

Pero como estas multiplicaciones equivalen a multiplicaciones por potencias de 10, son simples corrimientos del número multiplicado.

De esta manera:

1 0 0 0 x 1 1 0 1 1 0 1 0 1 = 1 1 0 1 1 0 1 0 1 0 0 0
1 0 x 1 1 0 1 1 0 1 0 1 = 1 1 0 1 1 0 1 0 10
1 x 1 1 0 1 1 0 1 0 1 = 1 1 0 1 1 0 1 0 1

Lo que se lleva simples corrimientos y sumas , que son mas rápidos.

b) Dadas las siguientes declaraciones (expresiones y operadores):

TRES = 3
entero: a = 5**,** b = 4
real: x = 5 **/** 2**,** y = 2**.**

¿son correctas las siguientes expresiones?, ¿por qué?. **(1,50 puntos)**

1) a **=** b **mod** TRES
2) 6 **div** TRES **<** TRES **mod** 6
3) TRES **+** b **-** 1 **<>** a **o** b **>= -** b ***** a **y** a**^** 2 **<=** 10
4) 1 **=** b **mod** a **div** TRES
5) **no (** x ***** a **>** y **/** b **)**

1) falso (el orden de actuación de los operadores es el siguiente: (mod) e (=)
2) verdadero (el orden de actuación de los operadores es el siguiente: (div), (mod) y (<)
3) verdadero (el orden de actuación de los operadores es el siguiente: signo menos (-), (^), (*), suma (+), resta (-), (>=), (<=), (<>), (y) y (o)
4) verdadero (el orden de actuación de los operadores es el siguiente: (mod) y (div) e (=)
5) falso (el orden de actuación de los operadores es el siguiente: (*), (/), (>) y (no)

P2. Desarrollar un algoritmo en lenguaje natural que permita calcular el número **e** para un número finito de términos, mediante la fórmula de Maclaurin se desarrolla la serie siguiente:

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!} + \ldots = \sum_{k=0}^{+\infty} \frac{1}{k!}$$

Hay que utilizar un procedimiento para el cálculo del factorial y disponer de todos los filtros necesarios para que no se produzca ningún error en la entrada de datos ni durante el cálculo. **(2,5 puntos)**

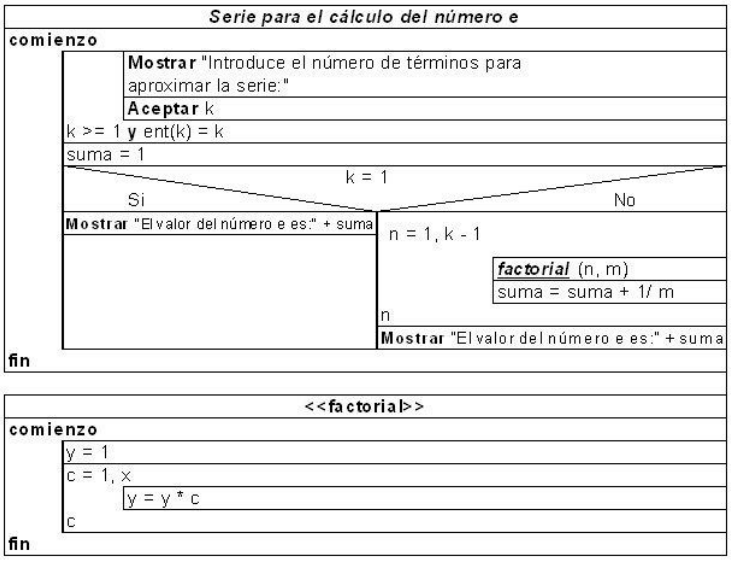
```
Algoritmo Serie para el cálculo del número e
variables
        entero: k (número de términos); n (contador del ciclo); m (valor del factorial); factorial (parámetros actuales del
        procedimiento)
        real: suma (solución)
comienzo
        repetir
                mostrar "Introduce el número de términos para aproximar la serie:"
                aceptar k
        hasta k >= 1 y ent(k) = k          //Comprobación para asegurar que el nº introducido es de tipo entero
        hacer suma ← 1
        si k = 1 entonces
                mostrar "El valor del número e es:" + suma
                sino
                        para n ← 1 hasta k – 1
                                llamar a factorial (n, m)
                                hacer suma ← suma + 1/ m
                        siguiente n
                mostrar "El valor del número e es:" + suma
        fin si
fin
----------Declaración del procedimiento----------
procedimiento factorial (x: entero; y: real)
variables
        entero: c  (contador del ciclo)
comienzo
        hacer y ←1
        para c ← 1 hasta x
                hacer  y ← y * c
        siguiente c
fin-procedimiento
```

a) Realizar la traza para k= 3. **(1 puntos)**

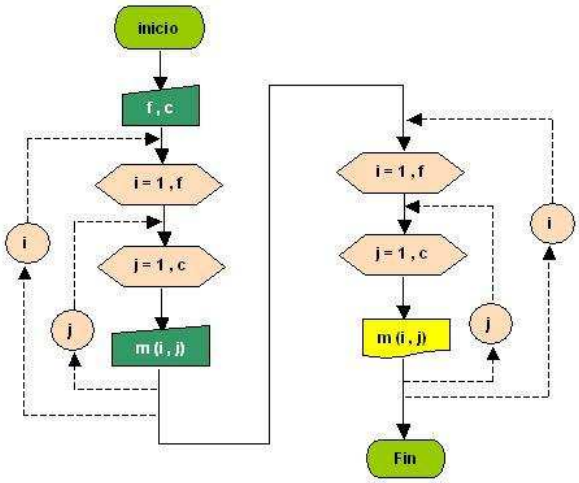
k	suma	n	suma
3	1	1→3	2
		2	2,5

y	c	y
1	1→1	1
	1→2	1
	2	2

b) Transformar el algoritmo desarrollado en diagrama N-S. **(1 puntos)**



P3. Dado el siguiente ordinograma trasformarlo a pseudocódigo. **(2,5 puntos)**




```
Algoritmo Ordinograma
Variables     entero: i , j (contadores); f (nº de filas); c (nº de columnas)
              Se desconoce el tipo de variable: m ( i , j ) (matriz de dimensión f , c)
comienzo
    aceptar f , c
    para i ← 1 hasta f
        para j ← 1 hasta c
            aceptar m ( i , j )
        siguiente j
    siguiente i
    para i ← 1 hasta f
        para j ← 1 hasta c
            escribir m ( i , j )
        siguiente j
    siguiente i
fin
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JUNIO 2.007 (Duración del examen: 3h)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

P1. **a)** Dado el siguiente algoritmo

```
Algoritmo Principal
variables
        entero: A; B; C
comienzo
    hacer A ← 2
    hacer B ← 3
        hacer C ← EfectoLateral (B)
    escribir A, B, C
        hacer C ← EfectoLateral (B)
    escribir A, B, C
fin

función EfectoLateral (X: entero): entero
comienzo
        hacer B ← X + 1
        EfectoLateral ← 2 * B
fin-función
```

Se pide:
Indicar cual de las siguientes soluciones sería la salida de datos de la última instrucción 'escribir' que se ejecuta, ¿por qué?. **(0,75 puntos)**

☐ a. 2, 3, 18

☐ b. 2, 3, 8

☐ c. **2, 5, 10**

☐ d. 2, 4, 8

Este es un ejemplo de lo que sucede si una variable global (B) modifica su valor dentro de un subalgoritmo. Al efectuar la traza se obtiene sucesivamente:

A	B	C	escribir	*C*	escribir
2	3	8	2, 4, 8	10	2, 5, 10

X	B	*EfectoLateral*
3	4	8
4	5	10

b) Dado el subalgoritmo

```
función Incorrecto (valor: entero): entero
variables
    entero: Resto
comienzo
    hacer Resto ← valor mod 10
fin-función
```

¿Qué hay equivocado en la siguiente función?, ¿por qué?. **(0,75 puntos)**

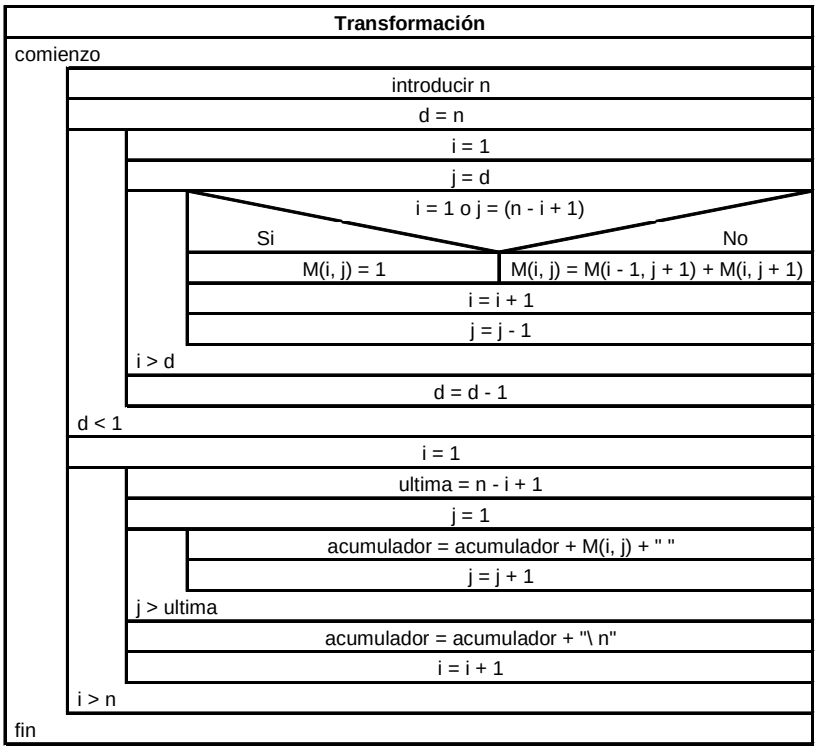
No hay asignación de un valor a la función *Incorrecto* que es el nombre de la función. Debe existir una instrucción por la que la función a través de su nombre devolverá un valor al algoritmo principal.

c) Pasar de binario a hexadecimal el número $1011111{,}110001_2$ **(0,75 puntos)**

La conversión entre binario y hexadecimal es igual al de la conversión octal y binario, pero teniendo en cuenta los caracteres hexadecimales, ya que se tienen que agrupar de 4 en 4.

Agrupando obtenemos el siguiente resultado:
$0101\ 1111,\ 1100\ 0100_2$
Sustituyendo según la tabla de conversión de binario a hexadecimal (Tema 1. Introducción a los ordenadores, láminas 23 y 24) logramos la conversión esperada:
$5F,\ C4_{16}$

P2. Transformar el siguiente algoritmo a pseudocódigo sustituyendo los bloques de control 'repetir' por la correspondiente instrucción de repetición con condición inicial sin que cambie el resultado final. **(1,50 puntos)**



Solución:

```
Algoritmo Pseudocódigo
variables      (Tipo de variable): i, j, n, M(i , j), d, ultima,  acumulador
comienzo
        aceptar n
        hacer d ← n
        mientras d >= 1
            hacer i ← 1
                  j ← d
             mientras i <= d
                si i = 1 ó j  = (n - i + 1)  entonces M(i , j) ← 1
                     si-no
                          hacer M(i , j)  ← M(i - 1 , j + 1) + M(i , j + 1)
                fin-si
                hacer i ← i + 1
                      j ← j – 1
            fin-mientras
            hacer d ← d - 1
        fin-mientras
        hacer i ← 1
        mientras i <= n
            hacer ultima ← n – i + 1
                  j ←1
            mientras j <= ultima
                hacer acumulador ←  acumulador + M(i , j) + " "
                      j ← j + 1
            fin- mientras
            hacer acumulador ←  acumulador + "\ n "
                  i ← i + 1
        fin-mientras
fin
```

P3. Desarrollar un algoritmo en lenguaje natural que permita calcular $\mathbf{X^n}$, donde:
X puede ser cualquier número real distinto de 0.
n puede ser cualquier entero positivo, negativo o nulo.

- Análisis del problema:

Si **n** = 0 entonces $\mathbf{X^n}$ valdrá 1, si **n** es positivo, entonces se multiplicará **X n** veces mediante un bucle en el que se vayan almacenando en una variable **producto**. Por medio de un contador controlaremos las veces que se ejecuta el bucle hasta alcanzar el valor de **n**. Si **n** es negativo, y **X** = 0 entonces no hay solución, si **X** <> 0 se procederá de la misma forma que si **n** es positivo, pero $\mathbf{X^n}$ será $1/ X^n$. Se diseñará el algoritmo según el análisis del problema, para ello usar las variables dadas en el análisis. **(2,50 puntos)**

```
Algoritmo Potencia de n
variables
        entero: n (exponente); i (contador del ciclo);
        real: X (base de la exponenciación); potencia (resultado de elevar X a n)
        lógico: sol (será falso si no hay solución)
comienzo
        aceptar X , n
        hacer sol ← verdad
        si n = 0 entonces
                hacer potencia ← 1
                sino
                        si n > 0 entonces
                            hacer potencia ← 1
                            hacer i ← 0
                            repetir
                                    hacer potencia ← potencia * X
                                    hacer i ← i + 1
                            hasta i = n
                            sino
                                si X = 0 entonces
                                    escribir "No hay solución."
                                    hacer sol ← falso
                                    sino
                                        hacer potencia ← 1
                                        hacer i ← 0
                                        repetir
                                            hacer potencia ← potencia * X
                                            hacer i ← i - 1
                                        hasta i = n
                                        hacer potencia ← 1 / potencia
                                fin-si
                        fin-si
        fin-si
        si sol entonces
            escribir "El valor de  X^n es:"; potencia
        fin-si
fin
```

a) Realizar la traza para a) n= 0, b) n= -1, c)n= 3 y X= 2 en los tres casos y d) n= -1 y X= 0. **(0,75 puntos)**

Caso	X	n	sol	potencia	i	sol	potencia	i	potencia	solución
a)	2	0	verdad	1						1
b)		-1		1	0		2	-1	1/2	0.5
c)		3		1	0		2	1		8
							4	2		
							8	3		
d)	0	-1				falso				No hay solución

P4. Diseñar un algoritmo en pseudocódigo, que lea una matriz de enteros de **m** filas x **n** columnas y calcule la suma de cada una de sus filas y columnas, mostrando por pantalla dichos resultados en 2 vectores, uno de la suma de las

filas vector **F** y otro de la suma de las columnas vector **C**. Diseñar el algoritmo usando las variables dadas en el enunciado. **(2,25 puntos)**

```
Algoritmo Suma de filas y columnas en una Matriz
variables
        entero: i (contador de filas); j (contador de columnas); m y n (dimensiones de la matriz)
          A(i, j) (matriz de m x n); F(i) (suma de elementos de filas); C(j) (suma de elementos de columnas)
comienzo
        escribir "Introduzca las dimensiones m y n de la matriz A:"
        aceptar m , n
        para i ← 1 hasta m
                para j ← 1 hasta n
                        aceptar A(i, j)
                siguiente j
        siguiente i
        para i ←1 hasta m
                para j ← 1 hasta n
                        hacer F(i) ← F(i) + A(i, j)
                siguiente j
                escribir "Fila " + i + ": " + F(i)
        siguiente i
        para j ← 1 hasta n
                 para i ← 1 hasta m
                        hacer C(j) ← C(j) + A(i, j)
                siguiente i
                escribir "Columna " + j + ": " + C(j)
        siguiente j
 fin
```

a) Realizar la traza para f = 3 ; c = 2 con A= $\begin{pmatrix} 1 & 3 \\ 4 & 0 \\ 3 & -1 \end{pmatrix}$. **(0,75 punto)**

m	n	i → m	j→n	A(i, j)	i → m	j→n	F(i)	j→n	i → m	C(j)	Suma de Filas	Suma de columnas
3	2	1→3	1→2	1								
			2	3								
		2	1→2	4								
			2	0								
		3	1→2	3								
			2	-1								
					1→3	1→2	1					
						2	4				4	
					2	1→2	4					
						2	4				4	
					3	1→2	3					
						2	2				2	
								1→2	1→3	1		
									2	5		
									3	8		8
								2	1→3	3		
									2	3		
									3	2		2

Código en Visual Basic del algoritmo anterior:

```
Dim m As Integer    'Dimensión de las filas de la matriz A
Dim n As Integer    'Dimensión de las columnas de la matriz A
Dim A() As Integer  'Matriz de dimensiones m x n
Dim i As Integer    'Contador de filas
Dim j As Integer    'Contador de columnas
Dim F() As Integer  'Vector suma de filas
Dim C() As Integer  'Vector suma de columnas

Private Sub Form_Load()
    m = Val(InputBox("Introduzca el número de filas de la matriz A:"))
```

```
    n = Val(InputBox("Introduzca el número de columnas de la matriz A:"))
    ReDim A(m, n)
    For i = 1 To m
        For j = 1 To n
            A(i, j) = InputBox("Elemento (" & i & ", " & j & ")=")
        Next j
    Next i
    ReDim F(m)
    ReDim C(n)
    For i = 1 To m
        For j = 1 To n
            F(i) = F(i) + A(i, j)
        Next j
        MsgBox "Fila " & i & ": " & F(i)
    Next i
    For j = 1 To n
        For i = 1 To m
            C(j) = C(j) + A(i, j)
        Next i
        MsgBox "Columna " & j & ": " & C(j)
    Next j
End Sub
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO SEPTIEMBRE 2.007 (Duración del examen: 2:30h)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

P1. Dado el siguiente algoritmo.

```
Algoritmo Cambio de base
Variables
        entero: n (número a cambiar de base); b (base de cambio comprendida entre 0 y 9)
comienzo
    hacer n ← 6
    hacer b ← 3
    llamar a conversion (n, b)
fin
```

```
procedimiento conversion (x, y: entero)        // y, entero comprendido entre 0 y 9
Variables
        entero: r (parte entera de d); c (resto entero de x / y)
        real: d (recoge el valor de la división entre x e y)
comienzo
        si x < y entonces        // Caso base solamente para n = 6 y b = 3
            mostrar "División " + x + "/" + y + "; " + "Cociente:" + r + " ; Resto:" + c
            sino                 // Caso general
                hacer d ← x / y
                hacer r ← ent(d)
                llamar a conversion (r, y)
                hacer c ← x - ent(d) * y
                mostrar "División " + x + "/" + y + "; " + "Cociente:" + r + " ; Resto:" + c
        fin-si
fin-procedimiento
```

¿Cuál de las siguientes posibles soluciones sería el resultado correcto del algoritmo?, ¿por qué?. **(1,50 puntos)**

a) División 2/3; Cociente: 1; Resto: 0
División 6/3; Cociente: 2; Resto: 0

b) **División 2/3; Cociente: 0; Resto: 0**
División 6/3; Cociente: 2; Resto: 0

c) División 2/3; Cociente: 1; Resto: 0
División 6/3; Cociente: 2; Resto: 1

d) División 2/3; Cociente: 2; Resto: 0
División 6/3; Cociente: 0; Resto: 0

Traza:

n (x)	b (y)	Llamada a procedimiento (n, b) (x, y)	Mostrar	d	r	Llamada a procedimiento (r, y)	c	Mostrar
6	3	6, 3	2/3; 0; 0	2	2	2, 3	0	6/3; 2; 0

Comprobación realizada con Visual Basic:

Solamente para n = 6 y b = 3, para otros valores puede no funcionar, ver nota al final de la página:
Con un formulario en vacío, copiar el código en la ventana 'Ver código' del Visual Basic.

```
Option Explicit
   Dim n As Integer     'número entero para cambiar de base
   Dim b As Integer     'número entero de la base de cambio
Private Sub Form_Load()
```

```
    n = Val(InputBox("Introduzca el número para cambiar de base ="))
    If n = "0" Then     'Cierra el InputBox y vuelve al formulario
       Exit Sub
    End If
    b = Val(InputBox("Introduzca la base de cambio ="))
    If b = "0" Then     'Cierra el InputBox y vuelve al formulario
       Exit Sub
    End If
    Call conversion(n, b)
    End                 'Cierra el formulario
End Sub
'Declaración del procedimiento recursivo que nos realizará el cambio de base
Sub conversion(x As Integer, y As Integer)
    Dim d As Single     'División de x entre y
    Dim c As Integer    'Resto
    Dim r As Integer    'Cociente
    If x < y Then       'Caso base para salir de la recursividad
       MsgBox "Número en base decimal: " & n & vbCrLf & "Nueva base: " & b & vbCrLf & "División " & x & "/" & y
                & "; " & "Cociente:" & r & " ; Resto:" & c
       Else             'Caso general
          d = x / y
          r = Int(d)
          Call conversion(r, y)
          c = x - Int(d) * y
          MsgBox "Número en base decimal: " & n & vbCrLf & "Nueva base: " & b & vbCrLf & "División " & x & "/" &
                   y & "; " & "Cociente:" & r & " ; Resto:" & c
    End If
End Sub
```

NOTA:
Comprobación realizada con Visual Basic, solamente para n = 6 y b = 3, para otros valores puede no funcionar, para que funcione hay que sustituir "If x < y Then" por "If x = 0 Then 'Caso base para salir de la recursividad con cualquier n y b", además si queremos mostrar los cocientes y restos correctos hay que sumar en el primer MsgBox a la variable r un uno:

```
MsgBox "Número en base decimal: " & n & vbCrLf & "Nueva base: " & b & vbCrLf & "División " & x & "/" &
        y & "; " & "Cociente:" & r + 1 & " ; Resto:" & c
```

Solución para el cambio de base: n = 6 y b = 3

División 0/3; Cociente: 1; Resto: 0
División 2/3; Cociente: 0; Resto: 2
División 6/3; Cociente: 2; Resto: 0

a) Dada la afirmación siguiente:
Cuando se llama a una función o a un procedimiento, el número de los parámetros actuales debe ser el mismo que el número de los parámetros formales y los tipos de los parámetros actuales deben coincidir con los tipos de los correspondientes parámetros formales, con una excepción: se puede asociar un parámetro actual de tipo real con un parámetro formal por valor de tipo entero.
¿Es correcta ésta afirmación?, ¿por qué?. **(0,75 puntos)**

No, ya que solamente se puede asociar un parámetro actual de tipo entero con un parámetro formal con valor de tipo real. Es decir no se puede almacenar un dato de tipo real en una variable de tipo entero, a la inversa sí que es posible.

b) ¿De qué clase pueden ser las variables que intervienen en algoritmos que usan **subalgoritmos**?. **(0,25 puntos)**
Define estas clases. **(0,50 puntos)**

Las variables que intervienen en un algoritmo con subalgoritmos pueden ser de dos clases: variables locales y variables globales.
- Las variables locales: son las que se utilizan en la definición de un subalgoritmo. Sólo tienen actuación dentro del subalgoritmo en el cual han sido declaradas y no son conocidas fuera de él.
- Las variables globales: están definidas en el algoritmo principal y tienen actuación tanto en el algoritmo principal como en los subalgoritmos que dependen de él.

P2. Desarrollar un algoritmo en lenguaje natural que permita calcular el número **e** para un número finito de términos, mediante la fórmula de Maclaurin se desarrolla la serie siguiente:

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{n!} + \ldots = \sum_{k=0}^{+\infty} \frac{1}{k!}$$

Hay que utilizar una función para el cálculo del factorial y disponer de todos los filtros necesarios para que no se produzca ningún error en la entrada de datos ni durante el cálculo. **(2 puntos)**

```
Algoritmo Serie para el cálculo del número e
variables
        entero: k (número de términos); n (contador del ciclo); factorial (argumento de la función factorial)
        real: suma (solución)
comienzo
        repetir
                mostrar "Introduce el número de términos para aproximar la serie:"
                aceptar k
        hasta k >= 1 y ent(k) = k          //Comprobación para asegurar que el nº introducido es de tipo entero
        hacer suma ← 1
        si k = 1 entonces
                mostrar  "El valor del número e es:" + suma
                sino
                    para n ← 1 hasta k - 1
                            hacer suma ← suma + 1/ factorial (n)
                    siguiente n
                mostrar "El valor del número e es:" + suma

        fin si
fin
----------------------------------------------------Declaración de la función----------------------------------------------------
función factorial (x = entero ) : entero
variables
        entero: fac (solución al factorial); c  (contador del ciclo)
comienzo
        hacer fac ←1
        para c ← 1 hasta x
                hacer  fac ← fac * c
        siguiente c
        factorial  = fac
fin-función
```

a) Realizar la traza para k= 3. **(0,75 puntos)**

k	suma	n	suma
3	1	1→3	2
		2	2,5

fac	c	fac
1	1→1	1
	1→2	1
	2	2

b) Transformar el algoritmo desarrollado en diagrama N-S. **(0,75 puntos)**

Serie para el cálculo del número e
comienzo
Mostrar "Introduce el número de términos para aproximar la serie:"
Aceptar k
k >= 1 **y** ent(k) = k
suma = 1
n = 1, k - 1
suma = suma + 1/ ***factorial*** (n)
n
Mostrar "El valor del número e es:" + suma
fin

<<factorial>>
comienzo
fac = 1
c = 1, x
fac = fac * c
c
factorial = fac
fin

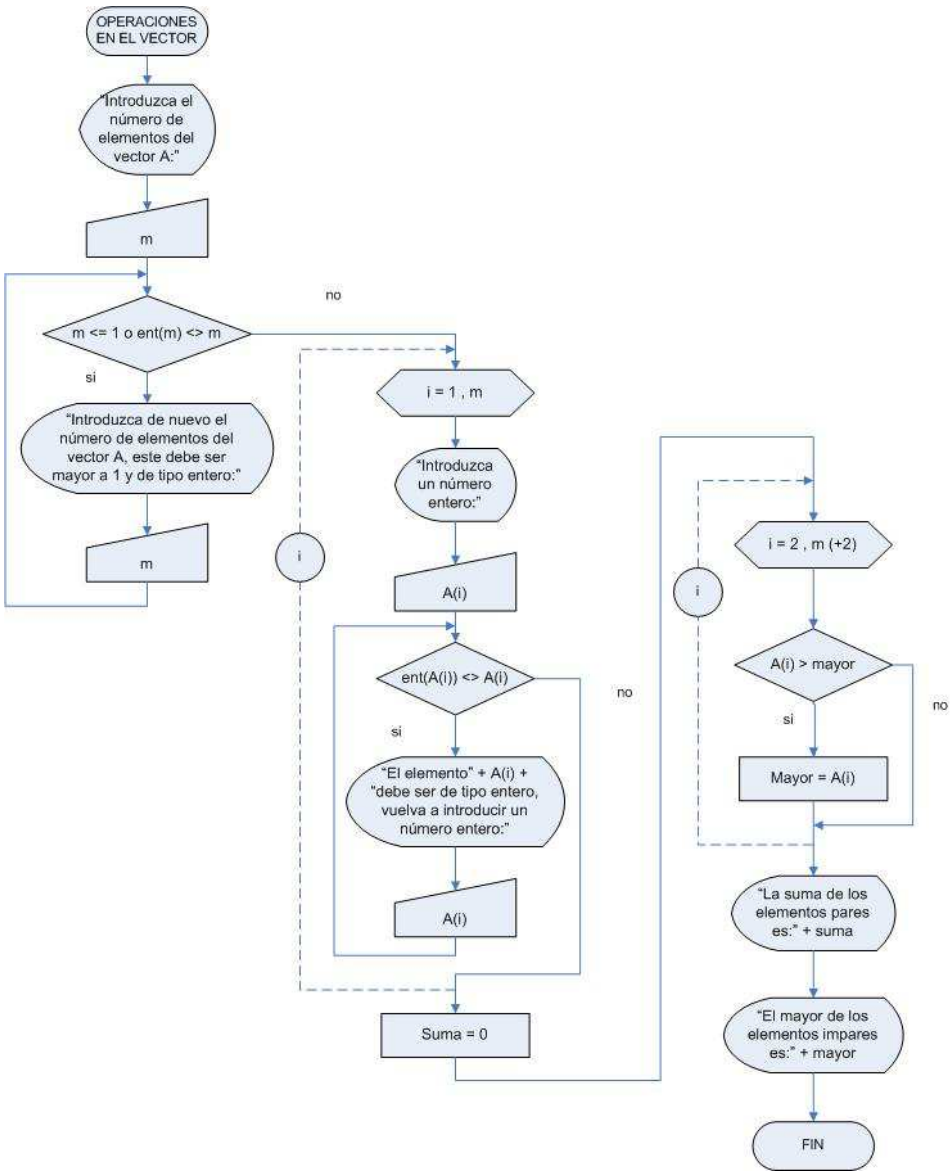
P3. Diseñar un algoritmo en pseudocódigo, que lea un vector **A** de **m** elementos de tipo entero y que calcule y muestre la **suma** de los elementos que ocupan posiciones pares, además de mostrar el elemento **mayor** de los que ocupan posiciones impares. Diseñar el algoritmo usando las variables dadas en el enunciado y disponer de todos los filtros necesarios para que no se produzca ningún error en la entrada de datos ni durante el cálculo. **(2 puntos)**

```
Algoritmo Operaciones en el vector
variables
        entero: i (contador de filas); m (número de elementos del vector); A(i) (vector de m elementos); suma (suma de elementos);
        mayor (elemento mayor)
comienzo
        mostrar "Introduzca el número de elementos m del vector A:"
        aceptar m
        mientras m <= 1 o ent(m) <> m
                mostrar "Introduzca de nuevo el número de elementos m del vector A, m deben ser mayor que 1 y de tipo entero:"
                aceptar m
        fin-mientras
        para i ← 1 hasta m
                mostrar "Introduzca un número entero:"
                aceptar A(i)
                mientras ent(A(i)) <> A(i)
                        mosrar "El elemento" + A(i) + "debe ser de tipo entero, vuelva a introducir un número entero:"
                        aceptar A(i)
                fin-mientras
        siguiente i
        hacer suma ← 0
        para i ←2 hasta m incremento 2
                hacer suma ← suma + A(i)
        siguiente i
        hacer mayor ← A(1)
        para i ← 1 hasta m
                si A(i) > mayor entonces
                        hacer mayor ← A(i)
                fin-si
        siguiente i
        mostrar "La suma de los elementos pares es: " +  suma
        mostrar "El mayor de los elementos impares es: " +  mayor
fin
```

a) Realizar la traza para m = 4 con A=$(-2 \quad 0 \quad 2 \quad 0)$. **(0,75 puntos)**

m	i	A(i)	suma	i	suma	mayor	i	mayor
4	1→4	-2	0	2→4	0	-2	1→4	-2
	2	0		4	0		3	2
	3	2						
	4	0						

b) Transformar el algoritmo desarrollado en diagrama de flujo. **(0,75 puntos)**


OPERACIONES EN EL VECTOR
"Introduzca el número de elementos del vector A:"
m
m <= 1 o ent(m) <> m
si
"Introduzca de nuevo el número de elementos del vector A, este debe ser mayor a 1 y de tipo entero:"
m
no
i = 1 , m
"Introduzca un número entero:"
A(i)
i
ent(A(i)) <> A(i)
si
"El elemento" + A(i) + "debe ser de tipo entero, vuelva a introducir un número entero:"
A(i)
Suma = 0
no
i = 2 , m (+2)
i
A(i) > mayor
si
Mayor = A(i)
no
"La suma de los elementos pares es:" + suma
"El mayor de los elementos impares es:" + mayor
FIN

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO DICIEMBRE 2.006 (MODELO 2) (Duración del examen: 3h)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

P1. Dadas las siguientes cuestiones, marcar con un aspa la solución correcta para cada una de las cuestiones, justificando la solución escogida.

1.- Cuando se realiza una llamada a un Subalgoritmo Procedimiento en pseudocódigo, ¿qué se indica entre paréntesis si es necesario? **(0,50 puntos)**

- ☐ a) Los parámetros de entrada
- ☐ b) Los parámetros de salida
- ☐ c) Los parámetros formales
- ☐ d) ***Los parámetros actuales***

El procedimiento realiza una tarea especifica que puede recibir cero, uno, o más valores del subalgoritmo que lo llama, y puede devolver cero, uno, o más valores a dicho algoritmo a través de una lista de parámetros, por lo tanto los parámetros pueden ser de entrada (comunican valores al procedimiento), de salida (proporcionan valores desde el procedimiento al algoritmo que lo llama) o de entrada y salida. En consecuencia, los procedimientos se llaman dentro de un algoritmo o subalgoritmo directamente por su nombre, añadiendo entre paréntesis la ***lista de parámetros actuales****, si los hay, que necesita para poder ejecutarse y en la declaración del procedimiento se añaden entre paréntesis la* ***lista de parámetros formales.***

2.- Para multiplicar por 2 un número en sistema binario: **(0,50 puntos)**

- ☐ a) Se añade un 1 al final.
- ☐ b) **Se añade un 0 al final.**
- ☐ c) Se permutan unos y ceros.
- ☐ d) Ninguna de las anteriores.

*Ya que 2 en binario equivale a 10 y si multiplicamos por ejemplo un número dado en binario como pueda ser el 11, siguiendo las reglas de 1 * 1 = 1, 1 * 0 = 0 *1 = 0 y 0 * 0 = 0. Obtenemos:*

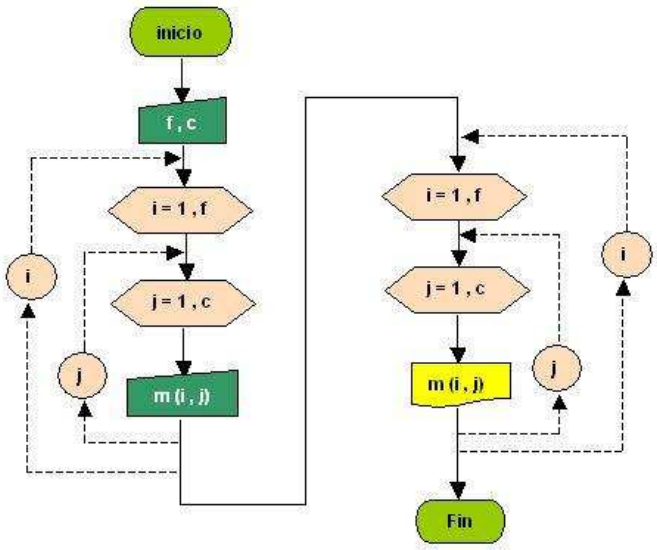
```
   11
 x 10
   00
+ 11
  110
```

3.- El lenguaje ensamblador es considerado: **(0,50 puntos)**

- ☐ a) De alto nivel.
- ☐ b) **De bajo nivel.**
- ☐ c) Intérprete.
- ☐ d) Lenguaje máquina.

Ya que es una mejora del lenguaje máquina, redactado según reglas mnemotécnicas. El programa que traduce estas reglas mnemotécnicas a lenguaje máquina se llama Ensamblador.

P2. Dado el siguiente ordinograma trasformarlo a pseudocódigo. **(1,50 puntos)**




```
Algoritmo Ordinograma
Variables     i , j = contadores
              f = nº de filas
              c = nº de columnas
              m ( i , j ) = matriz de dimensión f , c
comienzo
    aceptar f , c
    para i ← 1 hasta f
        para j ← 1 hasta c
            aceptar m ( i , j )
        siguiente j
    siguiente i
    para i ← 1 hasta f
        para j ← 1 hasta c
            imprimir m ( i , j )
        siguiente j
    siguiente i
fin
```

P3. Diseñar un procedimiento que obtenga la división entera y el resto de la misma utilizando únicamente los operadores matemáticos suma y resta, siguiendo los pasos indicados en el análisis del problema.

- Análisis del problema:

La división se puede considerar como una sucesión de restas. Se contarán cuántas veces se puede restar el *divisor* al *dividendo* y dicho contador será el *cociente*. Cuando no se pueda restar más sin que salga un número positivo, se tendrá el *resto*. **(2 puntos)**

Nota: considerar siempre que los valores pasados como parámetros son siempre positivos enteros y utilizar las variables dadas *divisor*, *dividendo*, *cociente* y *resto* únicamente para solucionar el problema.

```
procedimiento divisiónEntera (dividendo, divisor (E): entero ; cociente, resto (S): entero)
comienzo
    hacer cociente ← 0
    mientras dividendo >= divisor
        hacer dividendo ← dividendo – divisor
              cociente ← cociente + 1
    fin-mientras
    hacer resto ← dividendo
fin-procedimiento
```

a) Transformar el algoritmo a diagrama de Nassi-Shneiderman. **(0,75 puntos)**

<< divisiónEntera >>	
comienzo	
	cociente = 0
	dividendo >= divisor
	dividendo = dividendo – divisor
	cociente = cociente + 1
	resto = dividendo
fin	

b) Realizar la traza para dividendo = 7 y divisor = 3. **(0,75 puntos)**

7 div 3 = 2 cociente
7 mod 3 = 1 resto

cociente	dividendo	divisor	dividendo	cociente	resto
0	7	3	7 – 3 = 4	1	
			4 – 3 = 1	2	1

P4. Se pide diseñar un algoritmo en pseudocódigo, que sea capaz de averiguar si en una secuencia (vector *V*) dada e introducida por teclado de *n* números enteros y positivos, se encuentra un número *x* también entero, positivo e introducido por teclado. Se diseñará el algoritmo según el análisis del problema. **(2,75 puntos)**

Análisis del problema:

Se leerá el elemento del vector V(i) y se comparará con el número introducido x, así con cada uno de los elementos del vector hasta encontrar uno si lo hay idéntico. Para que el algoritmo sea efectivo, en el momento en que se encuentra un elemento idéntico al número x, ya no hará falta que se siga comparando cada elemento de la secuencia con el número x, al final se mostrará por pantalla si el número introducido por teclado existe en la secuencia o no. *Nota:* utilizar las variables dadas.

```
Algoritmo Búsqueda de un elemento en un vector
variables
        entero: n (nº de elementos del vector V); x (nº a buscar en la secuencia); V(i) (vector de n elementos);
        i (contador)
comienzo
    escribir "Introducir el nº de elementos de la secuencia:"
    aceptar n
    mientras n <= 0
        escribir "El nº de elementos de la secuencia ha de ser positivo, vuelva a introducir el nº de elementos:"
        aceptar n
    fin-mientras
    para i ← 1 hasta n
        escribir "Introduce un nº positivo para el elemento " + V(i) + ":"
        aceptar V(i)
        mientras V(i) < 0
            escribir "El nº del elemento " + V(i) + " ha de ser positivo, vuelva a introducir el nº del elemento:"
            aceptar V(i)
        fin-mientras
    siguiente i
    escribir "Introduce un nº positivo a buscar en la secuencia de números:"
    aceptar x
    mientras x < 0
       escribir "El nº a buscar el la secuencia ha de ser positivo, vuelva a introducir el nº a buscar:"
       aceptar x
    fin-mientras
    hacer i ← 1
    mientras x <> V(i) y i < n + 1
        hacer i ← i + 1   // Contador para salir del bucle mientras
    fin-mientras
    si i = n + 1  entonces
        escribir "El nº " + x + " no existe en la secuencia dada."
        si-no
            escribir "El nº " + x + " existe en la secuencia dada."
    fin-si
fin
```

a) Realizar la traza para n = 3, V (4, 3, 10), x = 3 y x = 2. **(0,75 puntos)**

- n = 3, V (4, 3, 10) y x = 3

n	i	V(i)	x	i	resultado
3	1 → 3	4			
	2	3			
	3	10	3	1	
				2	El nº 3 existe en la secuencia dada.

- n = 3, V (4, 3, 10) y x = 2

n	i	V(i)	x	i	resultado
3	1 → 3	4			
	2	3			
	3	10	2	1	
				2	
				3	
				4	El nº 2 no existe en la secuencia dada.

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO JUNIO 2.006 (Duración del examen: 3h)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

P1. Dada la siguiente estructura de repetición con condición inicial

```
mientras A <= B y B <> C
    hacer A ← A + 1
    hacer B ← B - 1
fin-mientras
```

Se pide:

a) Escribir la estructura de repetición con condición final equivalente a la anterior. **(0,75 puntos)**

```
repetir
    hacer A ← A + 1
    hacer B ← B - 1
hasta A > B y B = C
```

b) Dada la estructura

```
si B <> C y A <= B entonces
    hacer B ← B - 1
    hacer A ← A + 1
fin-si
```

¿Sería esta una estructura equivalente a la dada en el enunciado inicial?, ¿porqué?. **(0,75 puntos)**
No, porque no es una estructura de repetición, solamente es una estructura de selección.

P2. ¿Cuál sería el resultado correcto del siguiente algoritmo?, ¿porqué?. **(1,50 puntos)**

```
Algoritmo Cambio de base
Variables
        entero: (n = número a cambiar de base); (b = base de cambio comprendida entre 0 y 9)
comienzo
    hacer n ← 6
    hacer b ← 3
        llamar a conversion (n, b)
fin
```

```
procedimiento conversion (x, y: entero)        // y, entero comprendido entre 0 y 9
Variables
        entero: r (parte entera de d); c (resto entero de x / y)
        real: d (recoge el valor de la división entre x e y)
comienzo
        si x < y entonces        // Caso base solamente para n = 6 y b = 3
                mostrar "División " + x + "/" + y + "; " + "Cociente:" + r + " ; Resto:" + c
                sino             // Caso general
                        hacer d ← x / y
                        hacer r ← ent(d)
                        llamar a conversion (r, y)
                        hacer c ← x - ent(d) * y
                        mostrar "División " + x + "/" + y + "; " + "Cociente:" + r + " ; Resto:" + c
        fin-si
fin-procedimiento
```

Seleccionar la solución válida:

a) División 2/3; Cociente: 1; Resto: 0
   División 6/3; Cociente: 2; Resto: 0
b) División 2/3; Cociente: 0; Resto: 0
   División 6/3; Cociente: 2; Resto: 0
c) División 2/3; Cociente: 1; Resto: 0
   División 6/3; Cociente: 2; Resto: 1
d) División 2/3; Cociente: 2; Resto: 0
   División 6/3; Cociente: 0; Resto

Comprobación realizada con Visual Basic:

Solamente para n = 6 y b = 3, para otros valores puede no funcionar, ver nota al final de la página:
Con un formulario en vacío, copiar el código a la ventana de 'Ver código' del Visual Basic.

```
Option Explicit
   Dim n As Integer     'número entero a cambiar de base
   Dim b As Integer     'número entero de la base de cambio
Private Sub Form_Load()
   n = Val(InputBox("Introduzca el número a cambiar de base ="))
   If n = "0" Then     'Cierra el InputBox y vuelve al formulario
      Exit Sub
   End If
   b = Val(InputBox("Introduzca la base de cambio ="))
   If b = "0" Then     'Cierra el InputBox y vuelve al formulario
      Exit Sub
   End If
   Call conversion(n, b)
   End                 'Cierra el formulario
End Sub
'Declaración del procedimiento recursivo que nos realizará el cambio de base
Sub conversion(x As Integer, y As Integer)
   Dim d As Single     'División de x entre y
   Dim c As Integer    'Resto
   Dim r As Integer    'Cociente
   If x < y Then       'Caso base para salir de la recursividad
      MsgBox "Número en base decimal: " & n & vbCrLf & "Nueva base: " & b & vbCrLf & "División " & x & "/" & y
                & "; " & "Cociente:" & r & " ; Resto:" & c
      Else             'Caso general
         d = x / y
         r = Int(d)
         Call conversion(r, y)
         c = x - Int(d) * y
         MsgBox "Número en base decimal: " & n & vbCrLf & "Nueva base: " & b & vbCrLf & "División " & x & "/" &
                   y & "; " & "Cociente:" & r & " ; Resto:" & c
   End If
End Sub
```

NOTA:
Comprobación realizada con Visual Basic, solamente para n = 6 y b = 3, para otros valores puede no funcionar, para que funcione hay que sustituir "If x < y Then" por "If x = 0 Then 'Caso base para salir de la recursividad con cualquier n y b", además si queremos mostrar los cocientes y restos correctos hay que sumar en el primer MsgBox a la variable r un uno:

```
MsgBox "Número en base decimal: " & n & vbCrLf & "Nueva base: " & b & vbCrLf & "División " & x & "/" &
          y & "; " & "Cociente:" & r + 1 & " ; Resto:" & c
```

Solución para el cambio de base: n = 6 y b = 3

División 0/3; Cociente: 1; Resto: 0
División 2/3; Cociente: 0; Resto: 2
División 6/3; Cociente: 2; Resto: 0

P3. Desarrollar un algoritmo en lenguaje natural que permita mostrar por pantalla los números perfectos comprendidos en el intervalo de 0 a N sin hacer uso de las funciones internas matemáticas mod y div. **(2 puntos)**

Un número es perfecto si es igual a la suma de sus divisores[1], excluido el mismo. Por ejemplo:
6 es perfecto porque 6 = 1 + 2 + 3

[1] Un número es divisor de otro, si el primero divide exactamente al segundo, es decir, si el resto de la división del segundo por el primero es 0.

Solución A

```
Algoritmo Números perfectos
variables
        entero: RangoEstudio (límite superior del rango); n (nº a analizar); SumaDivisores (acumulador de los
                divisores de N); i (variable de ciclo y divisor); ok (variable tipo interruptor que toma los valores 0 y
                1); c (operación intermedia para la comprobación de si es divisor de n)

Comienzo
---------- Comprobación para que n sea siempre positivo y mayor que cero ----------
        hacer ok ← 0
        mientras ok = 0
            aceptar "Números perfectos contenidos en el intervalo entre 0 y "; RangoEstudio
            si n > 0 entonces
                hacer ok ← 1
                sino
                    escribir "El rango debe ser superior a 0."
            fin-si
        fin-mientras
---------- Comprobación para ver si n es un número perfecto ----------
        hacer SumaDivisores ← 0
        para i ← 1 hasta n – 1              // Se excluye el mismo
            hacer c ← n / i
            si Int(c) = c entonces          // Es divisor de n
                hacer SumaDivisores ← SumaDivisores + i
            fin-si
        siguiente i
---------- Salida de resultados ----------
        si SumaDivisores = i entonces
            escribir i + "es un número perfecto."
            si-no
                escribir i + "no es un número perfecto."
        fin-si
fin
```

a) Realizar la traza para N = 3 y N = 6. **(0,75 puntos)**

Traza de la solución A					
N	**SumaDivisores**	***I***	***c***	**SumaDivisores**	**SumaDivisores = *i***
3	0	1→2	3	1	
		2	1,5		
6	0	1→5	6	1	
		2	3	3	
		3	2	6	
		4	1,5		
		5	1,2		6 = 6

Solución B

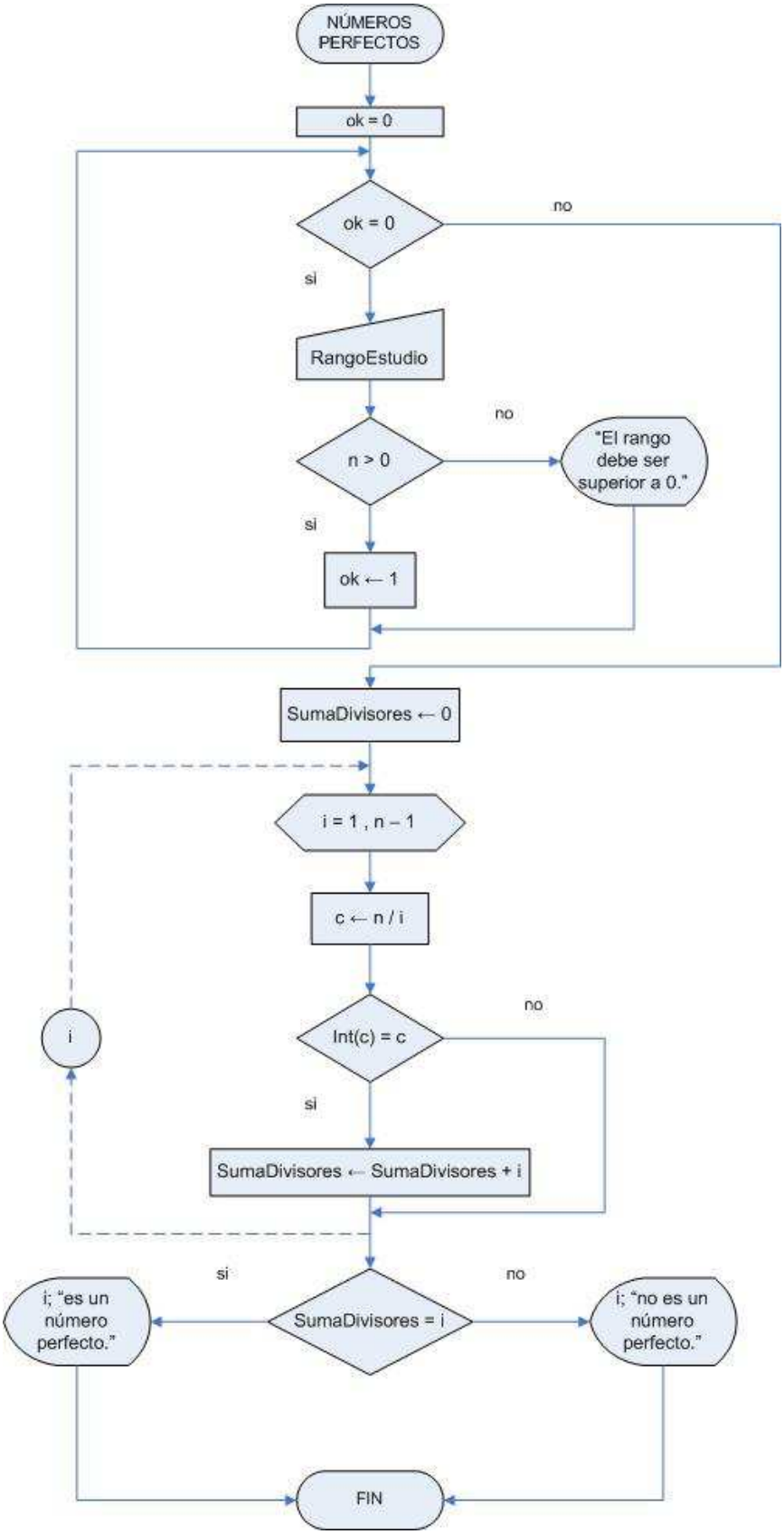
```
Algoritmo Números perfectos
variables
        entero: RangoEstudio (límite superior del rango); SumaDivisores (acumulador de los divisores de N); i
        (variable de ciclo y divisor); ok (variable tipo interruptor que toma los valores 0 y 1); CantNumPerfectos
        (contador de números perfectos); n (contador y número a comprobar); c (división de n entre i);
        NumerosPerfectos(CantNumPerfectos) (vector de números perfectos de dimensión CantNumPerfectos)
        alfanumérico: TextoResultado (contiene el texto a mostrar con el resultado)
Comienzo
-------------------------------- Comprobación para que n sea siempre positivo y mayor que cero --------------------------------
        hacer ok ← 0
        mientras ok = 0
            aceptar “Números perfectos contenidos en el intervalo entre 0 y ”; RangoEstudio
            si n > 0 entonces
                hacer ok ← 1
                sino
                    escribir "El rango debe ser superior a 0."
            fin-si
        fin-mientras
        hacer TextoResultado ← "Números perfectos entre 0 y " + RangoEstudio + ": "
        hacer CantNumPerfectos ← 0
        hacer n ← 1
        mientras n <= RangoEstudio
---------------------------------------- Comprobación para ver si n es un número perfecto ----------------------------------------
            hacer SumaDivisores ← 0
            hacer i ← 1
            mientras i <= n – 1              // Se excluye el mismo
                    hacer c ← n / i
                    si Int(c) = c entonces          ‘Es divisor de n
                        hacer SumaDivisores ← SumaDivisores + i
                    fin-si
            fin-mientras
------------------------------------------------------------ Salida de resultados ------------------------------------------------------------
            si SumaDivisores = n entonces
                    hacer CantNumPerfectos ← CantNumPerfectos + 1
                    hacer NumerosPerfectos(CantNumPerfectos) ← n
                    si CantNumPerfectos > 1 entonces
                        hacer TextoResultado ← TextoResultado + "; "
                        escribir TextoResultado
                    fin-si
                    hacer TextoResultado = TextoResultado + n
                    escribir TextoResultado
            fin-si
            hacer n = n + 1
        fin-mientras
        si CantNumPerfectos = 0 entonces
            escribir "No existen números perfectos en el intervalo dado."
        fin-si
fin
```

Comprobación realizada con Visual Basic:

Con un formulario en vacío, copiar el código a la ventana de 'Ver código' del Visual Basic.

```
Option Explicit
Dim RangoEstudio As Integer         'Valor introducido para el límite superior del rango
Dim TextoResultado As String        'Variable para mostrar acumulados los nº perfectos
Dim CantNumPerfectos As Integer     'Contador de números perfectos
Dim SumaDivisores As Integer        'Para comprobar si es un número perfecto
Dim NumerosPerfectos() As String    'Vector que contendrá los nº perfectos para ser mostrados
Dim ok As Integer                   'Variable de chequeo, puede ser 0 o 1
Dim n As Integer                    'Contador para controlar la lectura de todos los números del intervalo
Dim i As Integer                    'Contador para controlar la lectura de todos los números del intervalo excepto el que representa
el límite superior del rango
Dim c As Single                     'Para comprobar si es un número perfecto

Private Sub Form_Load()
   ok = 0
   While ok = 0
      RangoEstudio = Val(InputBox("Algoritmo que calcula e imprime los números perfectos contenidos en el intervalo entre
0 y:"))
      If RangoEstudio = "0" Then     'Cierra el InputBox y vuelve al formulario
         Exit Sub
      End If

      If RangoEstudio > 0 Then
         ok = 1
         Else
            MsgBox "El rango debe ser superior a 0."
      End If
   Wend
   TextoResultado = "Números perfectos entre 0 y " + Str(RangoEstudio) + ": "
   CantNumPerfectos = 0
   n = 1
   While n <= RangoEstudio
      SumaDivisores = 0
      i = 1
      While i <= n - 1
         c = n / i
         If Int(c) = c Then
            SumaDivisores = SumaDivisores + i
         End If
         i = i + 1
      Wend
      If SumaDivisores = n Then
         CantNumPerfectos = CantNumPerfectos + 1
         ReDim NumerosPerfectos(CantNumPerfectos)
         NumerosPerfectos(CantNumPerfectos) = n
         If CantNumPerfectos > 1 Then
            TextoResultado = TextoResultado + "; "
            MsgBox (TextoResultado)
         End If
         TextoResultado = TextoResultado + Str(n)
         MsgBox (TextoResultado)
      End If
      n = n + 1
   Wend
   If CantNumPerfectos = 0 Then
      MsgBox "No existen números perfectos en el intervalo dado."
   End If
End Sub
```

b) Transformarlo a diagrama de flujo. **(0,75 puntos)**
Diagrama de flujo de la solución A:



P4. Desarrollar un algoritmo en pseudocódigo, que lea los números *enteros positivos* de una lista A de N elementos y permita mostrar por pantalla en una nueva lista B los números primos ordenados de mayor a menor obtenidos de la lectura de la lista inicial A. **(2 puntos)**

Un número es primo si solamente es divisible por él mismo y la unidad.

```
Algoritmo Vector primo
variables
        entero: N (número de elementos del vector A(1…N); i (contador del ciclo); A(i) (vector de números enteros
                positivos); c (contador de números primos); j (contador para la comprobación de número primo);
                B(i) (vector de números primos ordenado de mayor a menor); k (variable auxiliar)
comienzo
        escribir "Introduce el número de elementos de que constará la lista A:"
        aceptar N
        mientras N <= 0
            escribir "El número de elementos mínimo de la lista ha de ser 1."
            aceptar N
        fin-mientras
        hacer c ← 0
        para i←1 hasta N
            escribir "Introduce un número entero positivo para el elemento " + i +":"
            aceptar A(i)
            mientras A(i) < 0
                escribir "El número debe ser positivo."
                aceptar A(i)
            fin-mientras
---------------------------------------- comprobación y guardado de nº primo en la lista B ----------------------------------------
            hacer j ← 2
            mientras j < A(i) y A(i) mod j <> 0
                hacer j ← j + 1
            fin-mientras
            si A(i) = j entonces        // Es primo
                hacer c ← c + 1
                hacer B(c) ← A(i)
            fin-si
        siguiente i
------------------------------------------------- ordenación de los elementos de la lista B -------------------------------------------
        para i←1 hasta c -1
            para k ← i +1 hasta c
                si B(k) > B(i) entonces
                    llamar a intercambiar (B(k), B(i))
                fin-si
            siguiente k
        siguiente i
---------------------------------Visualización ordenada de los números primos de la lista B ----------------------------------
            para i ← 1 hasta c
                visualizar B ( i )
            siguinte i
fin
```

```
Procedimiento intercambiar (x,y: entero)
variables
        entero: vp (variable auxiliar para el intercambio de elementos del vector)
comienzo
            hacer vp ← x
            hacer x ← y
            hacer y ← vp
fin-procedimiento
```

a) Realizar la traza para N = 4 ; A (2, 3, 5, 4).

Comprobación y guardado de nº primo en la lista B										
N	**c**	**i**	**A(i)**	**j**	**j < A(i) y A(i)mod j <> 0**	**j**	**A(i) = j**	**c**	**Es primo**	**B(i) = A(i)**
4	0	1→ 4	2	2	2 < 2 y 2mod2 = 0		2 = 2	1	si	2
		2	3	2	2 < 3 y 3mod2 = 1	3				
					3 < 3 y 3mod3 = 0		3 = 3	2	si	3
		3	5	2	2 < 5 y 5mod2 = 1	3				
					3 < 5 y 5mod3 = 2	4				
					4 < 5 y 5mod4 = 1	5				
					5 < 5 y 5mod5 = 1		5 = 5	3	si	5
		4	4	2	2 < 4 y 4mod2 = 0		4 = 2			

Ordenación de los elementos de la lista B							
i	**k**	**B(k) > B(i)**	**B(k) [x]**	**B(i) [y]**	**vp**	**x**	**y**
1 → 2	2 → 3	3 > 2	B(2) = 3	B(1) = 2	3	2	3
	3	5 > 3	B(3) = 5	B(1) = 3	5	3	5
2	3 → 3	3 > 2	B(2) = 3	B(1) = 2	3	2	3

Visualización de la lista B	
i	**B(i)**
1 → 3	5
2	3
3	2

Comprobación realizada con Visual Basic:

Con un formulario en vacío, copiar el código a la ventana de 'Ver código' del Visual Basic.

```
Option Explicit
Dim i As Integer        'contador del ciclo
Dim N As Integer        'número de elementos del vector A(1…N)
Dim A() As Integer      'vector de dimensiones i de números enteros positivos
Dim c As Integer        'contador de números primos y número de elementos del vector A(1…c)
Dim j As Integer        'contador para la comprobación de número primo
Dim B() As Integer      'vector de dimensiones i de números primos ordenado de mayor a menor
Dim k As Integer        'variable auxiliar

Private Sub Form_Load()
   N = InputBox("Introduce el número de elementos de que constará la lista A:")
   While N <= 0
      MsgBox "El número de elementos mínimo de la lista ha de ser 1."
      N = InputBox("Vuelva a introducir el número de elementos de que constará la lista A:")
   Wend
   c = 0
   ReDim A(N)
   For i = 1 To N
      A(i) = CInt(InputBox("Introduce un número entero positivo para el elemento " & i & ":"))
      While A(i) < 0
         MsgBox "El número debe ser positivo."
         A(i) = CInt(InputBox("Vuelva a introducir un número entero positivo para el elemento " & i & ":"))
      Wend
      'comprobación y guardado de nº primo en la lista B ----------------------------------------
      j = 2
      While j < A(i) And A(i) Mod j <> 0
         j = j + 1
      Wend
      If A(i) = j Then    'Es primo
         c = c + 1
         ReDim Preserve B(c)
         B(c) = A(i)
      End If
   Next i
   'ordenación de los elementos de la lista B ------------------------------------------------
   For i = 1 To c - 1
      For k = i + 1 To c
         If B(k) > B(i) Then
            Call intercambiar(B(k), B(i))
         End If
      Next k
   Next i
   'Visualización ordenada de los números primos de la lista B ---------------------------------
   For i = 1 To c
      MsgBox B(i)
   Next i
End Sub
```

```
'procedimiento intercambiar(x, y)
Sub intercambiar(x As Integer, y As Integer)
    Dim vp As Integer       'variable auxiliar para el intercambio de elementos del vector
    vp = x
    x = y
    y = vp
End Sub
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN TEÓRICO-PRÁCTICO SEPTIEMBRE 2.006 (Duración del examen: 3h)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

P1. Dadas las siguientes cuestiones, marcar con un aspa la solución correcta para cada una de las cuestiones, justificando la solución escogida.

1.- ¿Qué es un compilador? **(0,50 puntos)**

a) Un programa que traduce el lenguaje máquina a lenguaje de bajo nivel.
b) Un programa que transforma el programa escrito en lenguaje de alto nivel a lenguaje de bajo nivel.
c) Es un traductor que transforma cada instrucción del lenguaje de alto nivel a instrucciones de lenguaje máquina.
d) Un programa que transforma el programa escrito en lenguaje de bajo nivel a lenguaje máquina.

Ya que los compiladores se crearon para que la máquina (ordenador) interpretarán el lenguaje de alto nivel creado para facilitar al hombre la comunicación con la máquina, transformando los lenguajes de alto nivel a lenguaje máquina y no con lenguajes de bajo nivel o lenguaje máquina.

2.- El resultado de efectuar la operación 101101_2 * 2 en sistema octal es: **(0,50 puntos)**

a) 5A
b) 55
c) 90
d) 132

El 2 tanto en decimal como en octal equivalen en binario a 10, por lo tanto podemos optar por trabajar sumando o multiplicando para resolver la operación:

- Sumando:

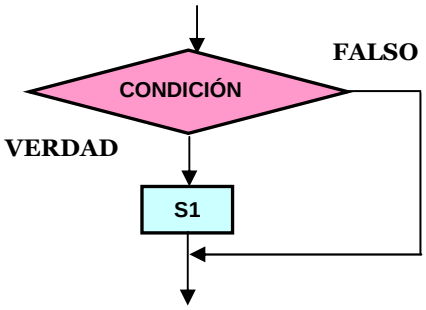
```
   1 11 1
   101101
+  101101
-------
  1011010   recordar que 1 + 1 = 0 y arrastramos 1, 0 + 1 = 1 + 0 = 0
```

- Multiplicación:

```
   101101
x      10
-------
  000000
 101101
-------
 1011010   entonces pasándolo a octal (grupos de tres cifras desde la derecha añadiendo ceros hasta
```

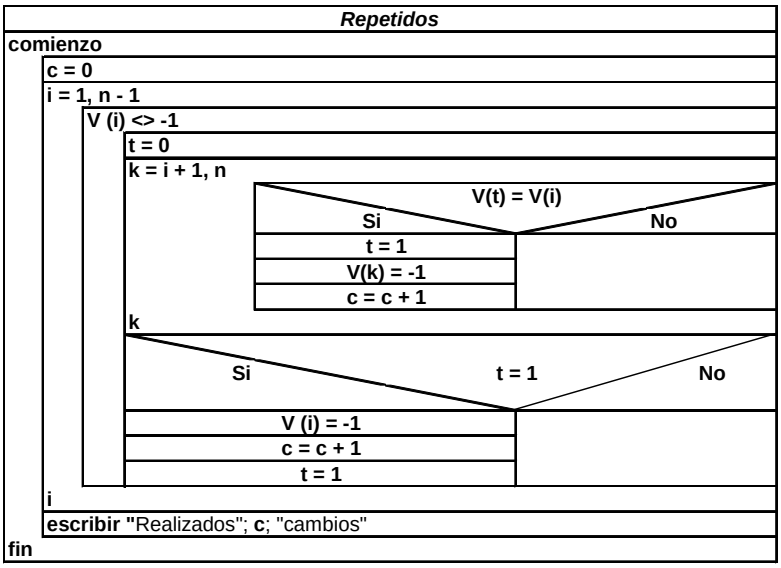
completar un grupo de tres con el último dígito de la izquierda) tenemos 001 011 010, que equivale a 13

3.- La siguiente estructura ¿de que tipo es? **(0,50 puntos)**



a) de repetición con condición inicial.
b) de selección sin cláusula sí-no.
c) de selección.
d) de repetición con más de dos alternativas.
Ya que no existe la posibilidad de optar a otra sentencia en caso de ser falsa.

P2. Dado el siguiente algoritmo en diagrama de Nassi-Shneiderman trasformarlo a pseudocódigo. **(1,50 puntos)**




```
Algoritmo Título
variables
        tipo: c; i; n; V (i); t; k; V (t); V (k)
comienzo
        hacer c ← 0
        para i ← 1 hasta n - 1
                mientras V(i) <> -1
                hacer t ← 0
            para k ← i + 1 hasta n
                si V(t) = V(i) entonces
                    hacer t ← 1
                                        hacer V(k) ← -1
                                        hacer c← c + 1
                                fin-si
            siguiente k
                        si t = 1 entonces
                                hacer V(i) ← -1
                                hacer c ← c + 1
                                hacer t ← 1
                        fin-si
                fin-mientras
        siguiente i
    escribir “Realizados ”; c;” cambios.”
fin
```

P3. Diseñar una función no recursiva en pseudocódigo para obtener la sucesión de Fibonacci (1, 1, 2, 3, 5, 8, 13, 21, 34,…) para n términos, siguiendo los pasos indicados en el análisis del problema. **(2 puntos)**

Análisis del problema:

Excluidos los 2 primeros términos de la sucesión, cualquier otro término se obtiene por la suma de los dos términos inmediatamente anteriores ($F_n = F_{n-1} + F_{n-2}$).

F será el nº de Fibonacci obtenido por la suma de los dos nº anteriores, que se indican con F1 y F2.

F = F1 + F2

Inicialmente F1 = 1 y F2 = 1 y mediante un ciclo, se calculan los otros términos hasta el enésimo.
A cada paso del ciclo se calcula el valor de un nuevo nº de Fibonacci mediante la relación F = F1 + F2 y a F1 se le asigna el valor de F2 y a F2 el valor de F anteriormente obtenido. *Nota:* utilizar las variables dadas.

```
función fibonacci (n = entero): entero      'el parámetro n de entrada provendrá del algoritmo principal y nos
                                            indica el nº de términos de la sucesión
variables
        entero: i (contador del ciclo); F1, F2, F (nº de Fibonacci)
comienzo
        hacer   F ← 1
        F1 ← 1
        F2 ←1
        para i ← 2 hasta n - 1
                hacer   F1 ← F2
                        F2 ← F
                        F ← F2 + F1
                        fibonacci ← F
        Siguiente i
fin-función
```

a) Realizar la traza para n = 6.

F	F1	F2	i	F1	F2	F	*fibonacci*
1	1	1	2 → 6	1	1	2	2
			3	1	2	3	3
			4	2	3	5	5
			5	3	5	8	8

La solución obtenida quedaría de la siguiente forma:

Por ejemplo, el algoritmo principal nos proporcionaría: 1, 1,
Y la función: 2, 3, 5, 8
El algoritmo principal deberá mostrar: 1, 1, 2, 3, 5, 8

P4. Una matriz A de la forma:

$$A = \begin{pmatrix} a_{11} & a_{12} & \ldots\ldots & a_{1n} \\ a_{21} & a_{22} & \ldots\ldots & a_{2n} \\ \cdot & \cdot & & \\ \cdot & \cdot & & \\ a_{m1} & a_{m2} & \ldots\ldots & a_{mm} \end{pmatrix}_{m \times n}$$

es simétrica si m = n y se cumple que A (i, j) = A (j, i) para 1 < = i < = n y 1 < j < = m. Se pide diseñar un algoritmo en pseudocódigo que lea una matriz de m x n y determine si es o no simétrica siguiendo los pasos indicados en el análisis del problema. **(3,50 puntos)**

Análisis del problema:

Para que el algoritmo sea efectivo solamente hay que comprobar que los elementos de la matriz que están por debajo de la diagonal principal por ejemplo, sin incluir a la diagonal principal sus simétricos sean iguales. Por ejemplo:

Se leerá el elemento de la matriz A(2, 1) y se comparará con el elemento de la matriz A(1, 2), así con cada uno de los elementos de la matriz que están por debajo de la diagonal principal, si un elemento no es simétrico ya no hará falta que se siga comparando cada elemento, al final se mostrará por pantalla si la matriz dada es simétrica o no. *Nota:* utilizar las variables dadas.

```
Algoritmo Matriz simétrica
variables
        entero: m (filas de la matriz A); n (columnas de la matriz A); f, c (contadores); A (f, c) (matriz A de dimensiones f, c)
        alfanumérico: sw  (testigo)
comienzo
        aceptar m, n
        mientras m < 0 y n < 0 hacer
            aceptar m, n
        fin-mientras
        para f ← 1 hasta m
                para c ← 1 hasta n
                    aceptar A(f, c)
                siguiente c
        siguiente f
        si m <> n entonces
            visualizar "No simétrica"
            si-no
                si m = n y m = 1 entonces
                    visualizar "Es simétrica"
                    si-no
                        hacer sw ← "verdad"
                        hacer f ← 2
                        mientras f <= m y sw = "verdad"
                            hacer c ← 1
                            mientras c <=  f – 1 y sw = "verdad"  hacer
                                si A(f, c) <> A(c, f) entonces
                                    hacer sw ← "falso"  // Testigo para no seguir comprobando
                                fin-si
                                hacer c ← c + 1   // Contador para salir del bucle mientras
                            fin-mientras
                            hacer f ← f + 1      // Contador para salir del bucle mientras
                        fin-mientras
                        si sw = "verdad"  entonces
                            visualizar "Simétrica"
                            si-no
                                visualizar "No simétrica"
                        fin-si
                fin-si
        fin-si
fin
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN PRÁCTICO DICIEMBRE 2.005 (Duración del examen 2h30m)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

P1. **a)** Realizar un algoritmo en pseudocódigo que calcule e^a, para un valor dado de a y para k términos.

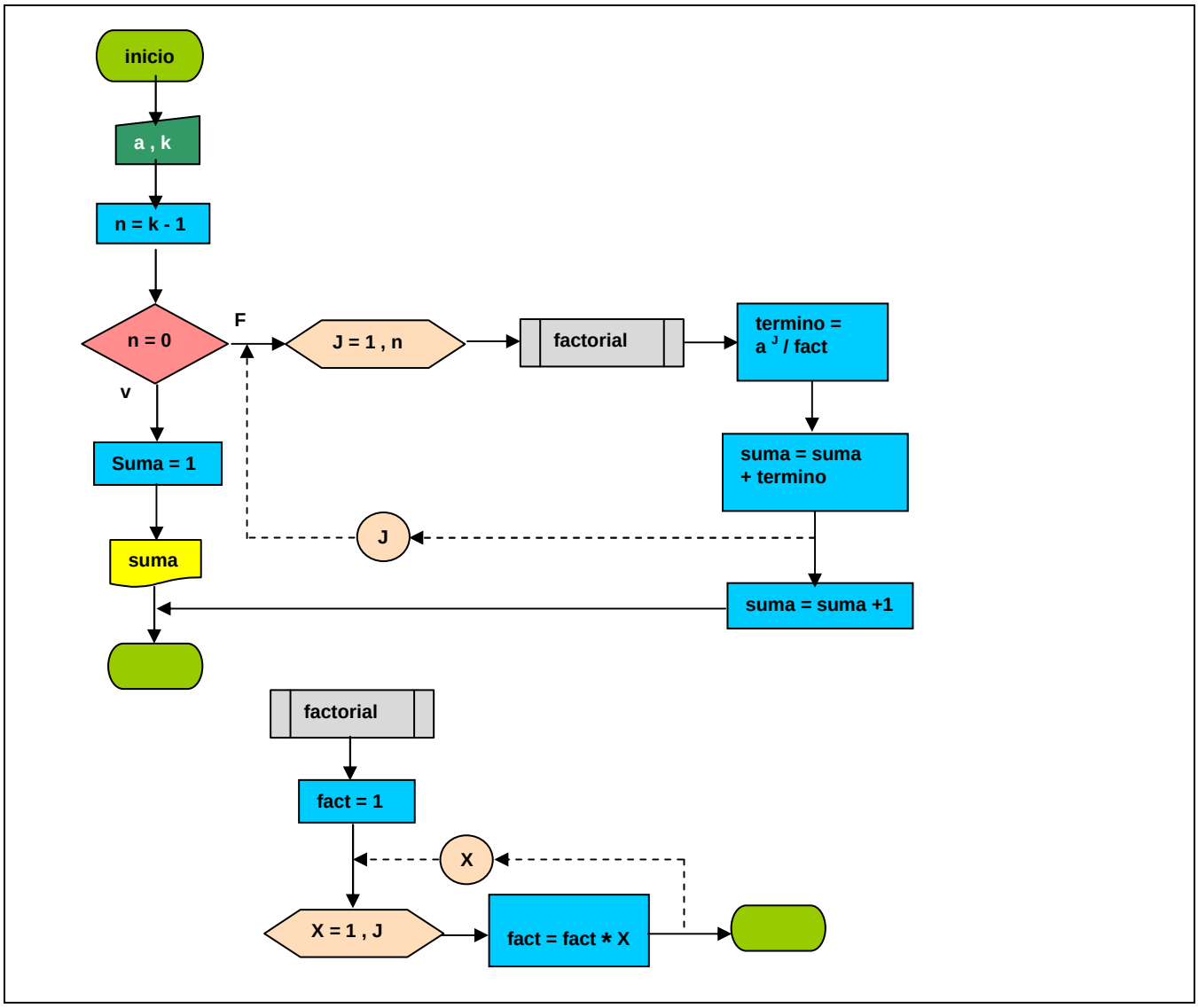
$$e^a = 1 + a + a^2/2! + a^3/3! + ... + a^k/k!$$

utilizando una función para el cálculo del factorial. **(3,0 puntos)**

```
Algoritmo cálculo de e^a
variables    real: a (número); suma (acumulador); J (contador)
             entero: n (número); k (términos de la sucesión); facto (valor del factorial); término (resultado de una
operación)
comienzo
        aceptar a , k
        hacer n ← k -1
        si n = 0 entonces  suma←1
            si-no
                para  J ←1 hasta n
                        hacer   facto ←  factorial ( J )
                                término ← a^J / facto
                                suma← suma + termino
                 siguiente J
                hacer suma← suma + 1
        fin-si
        mostrar suma
fin
```

```
función factorial  ( x : entero ) : entero
variables entero: fac (valor del factorial); i (contador
comienzo
        hacer fac ← 1
        para i ← 1 hasta x
              fac ←  fac * i
        siguiente i
        factorial  ← fac
fin-función
```

b) Escribir el ordinograma correspondiente del anterior apartado. **(1,0 puntos)**



P2. a) Desarrollar un algoritmo en lenguaje natural que permita escribir un cuadrado latino . **(2,5 puntos)**

Un **cuadrado latino** es una matriz M cuadrada de orden n, en la que su primera fila contiene los n primeros números naturales y cada una de las n-1 filas siguientes, la rotación de la fila anterior un lugar a la derecha.

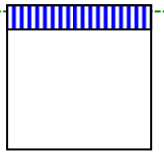
Ejemplo: para n = 5

1	2	3	4	5
5	1	2	3	4
4	5	1	2	3
3	4	5	1	2
2	3	4	5	1

(n x n)

FORMA ELEGANTE

```
Algoritmo Cuadrado latino
variables    (falta indicar el tipo de variables)
             i , j = contadores de filas y columnas respectivamente
             n = nº de filas y de columnas de la matriz M
             M(i , j) = matriz de dimensión n , n
             Num = contador de números naturales

Comienzo
---------- Introducción de la fila superior----------
        aceptar "Introduzca la dimensión del cuadrado latino"; n
        hacer Num = 0
        para j ← 1 hasta n
              hacer Num ← Num + 1
              hacer M(1 , j) ← Num
        siguiente j
---------- Introducción de la 1ª columna excepto su primer elemento y el resto del cuadrado ----------
        para i ← 2 hasta n
            para j ← 1 hasta n
                si j = 1 entonces
                    hacer M(i , j) ← M(i – 1 , n)
                si-no
                    hacer M(i , j) ← M(i – 1 , j – 1)
                fin-si
            siguiente j
        siguiente i
---------- Visualización----------
        para i ← 1 hasta n
            para j ← 1 hasta n
                  mostrar M(i , j) +  "    "          // se añade un espacio en blanco
            siguiente j
            mostrar "                                "          // se añade un línea en blanco
        siguiente i
fin
```



b) Calcular la traza para una matriz 5x5 (n=5) **(1,0 puntos)**

n	i	j	Num	M(i , j)	M(i - 1, n)	M(i , j)	M(i - 1, j -1)	M(i , j)	
5	1	1	1	1					i1
		2	2	2					i2
		3	3	3					i3
		4	4	4					i4
		5	5	5					i5
	2	1			(1 , 5)	5			
		2					(1 , 1)	1	
		3					(1 , 2)	2	
		4					(1 , 3)	3	
		5					(1 , 4)	4	
	3	1			(2 , 5)	4			
		2					(2 , 1)	5	
		3					(2 , 2)	1	
		4					(2 , 3)	2	
		5					(2 , 4)	3	
	4	1			(3 , 5)	3			
		2					(2 , 1)	4	
		3					(2 , 2)	5	
		4					(2 , 3)	1	
		5					(2 , 4)	2	
	5	1			(4 , 5)	2			
		2					(2 , 1)	3	
		3					(2 , 2)	4	
		4					(2 , 3)	5	
		5					(2 , 4)	1	
1	1			1					
	2			2					
	3			3					
	4			4					
	5			5					
2	1			5					
	2			1					
	3			2					
	4			3					
	5			4					
3	1			4					
	2			5					
	3			1					
	4			2					
	5			3					
4	1			3					
	2			4					
	3			5					
	4			1					
	5			2					
5	1			2					
	2			3					
	3			4					
	4			5					
	5			1					

CÁLCULO DE LOS VALORES

VISUALIZACIÓN DE LOS VALORES

j1	j2	j3	j4	j5
1	2	3	4	5
5	1	2	3	4
4	5	1	2	3
3	4	5	1	2
2	3	4	5	1

n=1 (i,j)	n=2 (i,j)	n=3 (i,j)	n=4 (i,j)	n=5 (i,j)
1	2	3	4	5
(1,5)	(1,1)	(1,2)	(1,3)	(1,4)
(2,5)	(2,1)	(2,2)	(2,3)	(2,4)
(3,5)	(3,1)	(3,2)	(3,3)	(3,4)
(4,5)	(4,1)	(4,2)	(4,3)	(4,4)

Traducción al lenguaje de programación del editor de plantillas de Arquímedes:
(Se ha usado un guión antes de imprimir en la zona resumen)

```
#Introducción de la fila superior
# -------------------------------------------------------------------------
n = dimension
Num = 0
f = 1
c = 1
WHILE c <= n
   Num = Num + 1
   M(f,c) = Num
   c+=1
END
#Introducción de la 1ª columna excepto su primer elemento y el resto del cuadrado
# -------------------------------------------------------------------------
f = 2
WHILE f <= n
   c = 1
```

```
    WHILE c<= n
        IF c=1 THEN
            M(f,c) = M(f-1,n)
        ELSE
            M(f,c) = M(f-1,c-1)
        END
        c += 1
    END
    f+= 1
END
#Impresión
# ------------------------------------------------------------------------
f = 1
WHILE f <= n
    c = 1
    WHILE c <= n
        cuadrado = cuadrado + M(f,c) + "    " # se añade tabulador
        c += 1
    END
    cuadrado = cuadrado + "\n" # se añade salto de línea
    f += 1
END
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

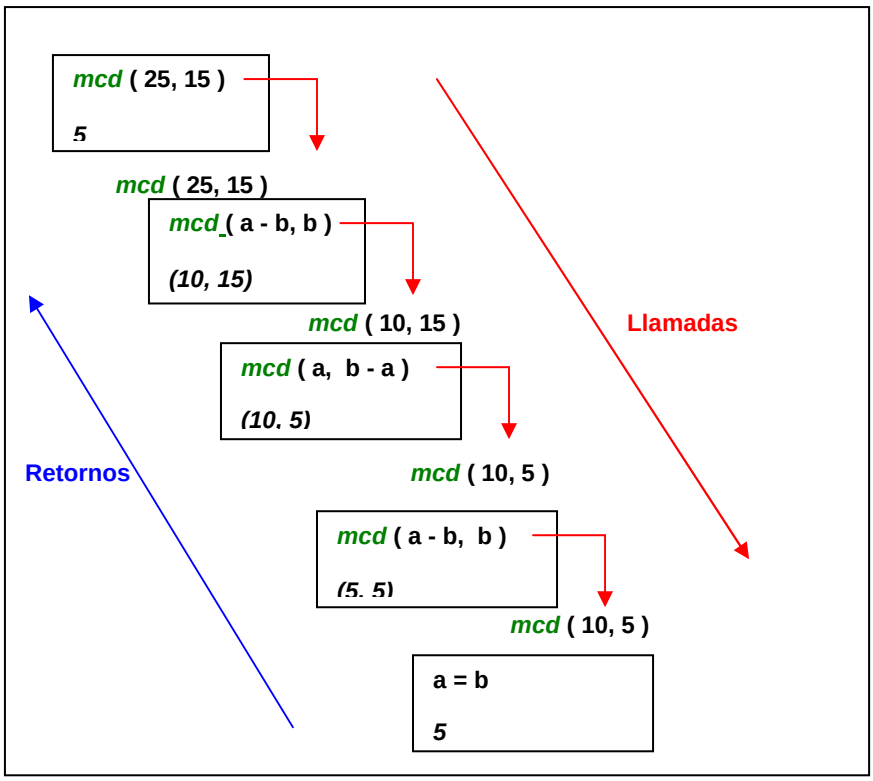
EXAMEN PRÁCTICO JUNIO 2.005 (Duración del examen 2h40m)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

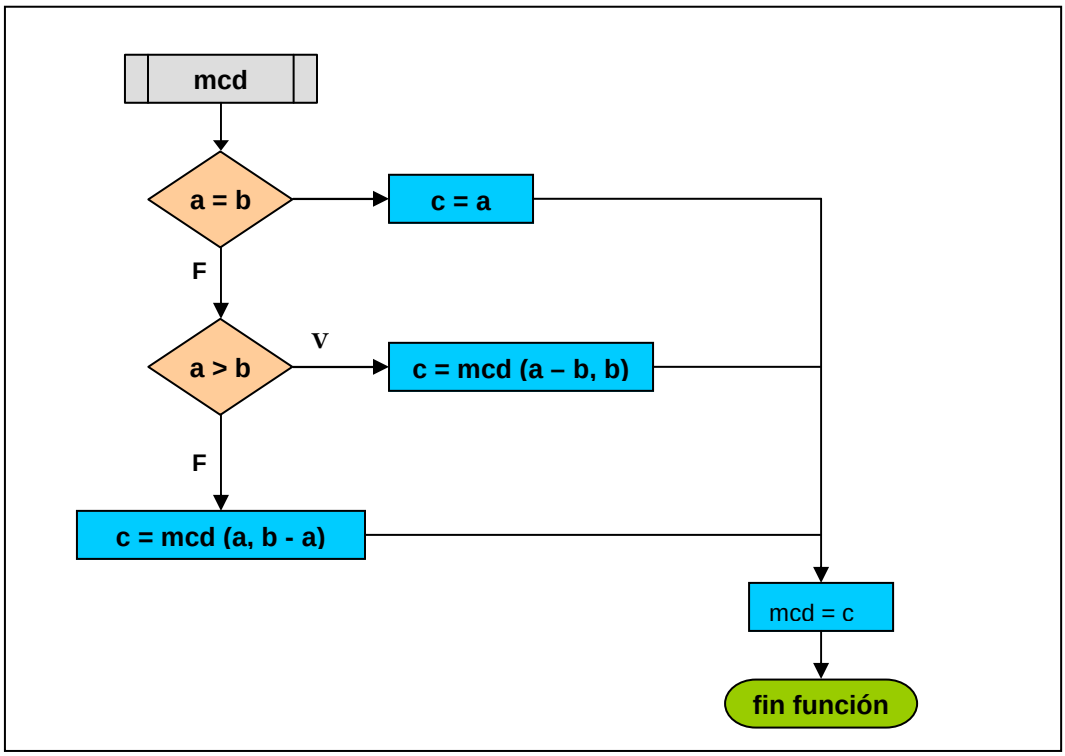
P1. Crear un subalgoritmo en pseudocódigo que permita calcular el máximo común divisor m.c.d. sin usar el Teorema de Euclides, utilizando para ello una función recursiva. **(1 punto)**

```
Función mcd ( a, b : entero ) : entero
variable
        entero: c(máximo común divisor)
comienzo
    si a = b entonces                          //caso base
        hacer c ← a
    sino
        si a>b entonces                        //caso general o recursivo
            hacer c ← mcd ( a - b, b)
        sino
            hacer c ← mcd ( a, b - a)
        fin-si
    fin-si
    hacer mcd ← c
fin-función
```

a) Realizar las llamadas y retornos para a = 25 , b = 15 **(0,5 punto)**



b) Trasformar a ordinograma la función recursiva. **(0,5 punto)**



P2. Dada la siguiente tabla de datos:

				Código	Descrip.
				↓	↓
				n_1	n_2
UNIDADES DE OBRA	U01	→	f_1		
	U02	→	f_2		
	U03	→	f_3		
		⋮			
		i			P(i,j)
		⋮			
	U_n	→	f_n		

= **P**(f,n)

Medición	PV	Cert.1	Cert.2				Cert.$_n$
↓	↓	↓	↓	...	j	...	↓
c_1	c_2	c_3	c_4				c_n
					M(i,j)		

= **M**(f,c)

donde las filas están representadas por la letra " f ", las cuales contienen las distintas unidades de obra de un presupuesto parcial. En las tablas P(f,n) y M(f,c) las columnas vienen representadas por las letras "n" y "c", conteniendo P(f,n) los códigos de las unidades de obra con sus respectivas descripciones y M(f,c) las cantidades certificadas a origen en cada certificación de cada unidad de obra a excepción de la columna c1 y c2, las cuales contienen la medición contratada y el precio de venta (PV) del contrato respectivamente. Se pide desarrollar un algoritmo en pseudocódigo que permita:

a) Calcular el importe certificado a origen (Io) de la tercera certificación de cada unidad de obra. Además se deberá calcular el importe total certificado a origen (ITo) del presupuesto parcial de esa tercera certificación. **(1 punto)**

b) Calcular y escribir la cantidad certificada parcial (Cp) de la última certificación de cada unidad de obra con su respectivo código y descripción. **(1 punto)**
NOTA: Cantidad certificada parcial (Cp) = Certificación a origen n [Cp(n)] - Certificación a origen n-1 [Cp(n-1)]

c) Calcular el exceso de obra (Eo) de aquellas unidades de obra que cumplan la condición, e identificar la certificación junto al código de la unidad de obra implicada con su cantidad excedida.
NOTA: Exceso de obra (Eo) = Cantidad certificada a origen (Co) – Medición contratada (M)
Condición para que exista Exceso de obra (Eo) → Cantidad certificada a origen (Co) > Medición contratada (M) **(1 punto)**

```
Algoritmo Certificación
variables   (falta indicar el tipo de variables)
            f = filas de la matriz P y M
            n = columnas de la matriz P
            c = columnas de la matriz M
            i, j = contadores
            P(i, j) = Matriz P de dimensiones f, n
            M(i, j) = Matriz M de dimensiones f, c
            Io = Importe certificado a origen
            Ito = Importe total certificado a origen del presupuesto
            Cp = Cantidad certificada parcial
            Eo = Exceso de obra
 comienzo
      escribir "Nº de unidades de obra:"// Lectura de datos para el algoritmo y solución al apartado a)
      leer f
      escribir "Nº de certificaciones:"
      leer c
      hacer c ← c + 2
      para i ← 1 hasta f
            escribir "Datos de la unidad de obra nº:" + i
            escribir "Código:"
            leer P(i,1)
            escribir "Descripción:"
            leer P(i,2)
            escribir "Medición:"
            leer M(i,1)
            escribir "PV:"
            leer M(i,2)
            para j ← 1 hasta c
                  escribir "Certificación nº:" + j
                  leer M(i,j+2)
            siguiente j
      siguiente i
      hacer Ito ← 0
      para i ← 1 hasta f
            hacer Io ← M(i,5) * M(i,2)
            hacer Ito ← Ito + Io
      siguiente i
      para i ← 1 hasta f          // Solución al apartado b)
            hacer Cp ← 0
            hacer j ← c
            hacer Cp ← M(i,j) - M(i,j -1)
            escribir P(i,1) + "  " + P(i,2) + " = " + Cp
      siguiente i
      para i ← 1 hasta f          // Solución al apartado c)
            hacer Eo ← 0
            para j ← 3 hasta c
                  si M(i,j) > M(i,1) entonces
                        hacer Eo ← M(i,j) - M(i,1)
                        escribir "Certificación nº" + j + "con exceso de obra" + P(i,1) + " = " + Eo
                  fin si
            siguiente j
      siguiente i
fin
```

P3. El **método de la burbuja** para ordenar vectores consiste en comparar un elemento y su consecutivo a lo largo del vector, si estan ordenados se repite el proceso con el elemento siguiente, si no lo están se intercambian sus posiciones (mediante un procedimiento) y se compara con los elementos ordenados anteriormente. Se llama de la burbuja porque en la ordenación los elementos de menor valor (más ligeros) suben en la lista como burbujas a la superficie de un líquido.
Dado el algoritmo:

```
Algoritmo burbuja
Variables  (falta indicar el tipo de variables usadas)
            dimen=dimensión de la tabla
            elem=elemento de la tabla
            I,K=contadores
Comienzo
    aceptar dimen
     ----------introdocir los elementos de la tabla ----------
    Para i ← 1 hasta dimen
            aceptar elem ( i )
    siguinte i
     ---------- ordenar los elementos ----------
    Para i ← 1 hasta dimen - 1
            Para k ← i hasta 1 decreciendo en 1
                Si elem( k + 1 ) < elem ( k ) entonces
                    hacer intercambiar (elem( K+1), elem( K))
                fin-si
            siguiente k
    siguiente i
Fin.
```

```
Procedimiento intercambiar (x,y: real)
Variables
        entero: vp  (variable auxiliar)
comienzo
            hacer vp ← x
            hacer x ← y
            hacer y ← VP
fin-procedimiento
```

Se pide:

a) Diseñar un **refinamiento** del algoritmo anterior que optimice el funcionamiento del mismo **(1,5 puntos)**

Se debe de introducir la condición de que si el elemento k+1 es menor al k, pare de comprobar para ese valor de i y pase al siguiente. Para ello hay que emplear estructuras de repetición con contador, proponiéndose dos soluciones, una buena y otra óptima.

A. SOLUCIÓN VALORADA EN 1 PUNTO:

```
---- ordenar los elementos -----
para i ← 1 hasta dimen-1
    hacer k ← i
    repetir
        si elem(k+1)<elem(k) entonces
            hacer intercambiar (elem(k+1),elem(k))
        fin-si
        hacer k ← k-1
    hasta elem(k+1)>=elem(k)
siguiente i
```

B. SOLUCIÓN VALORADA EN 1,5 PUNTOS:

```
---- ordenar los elementos -----
para i ← 1 hasta dimen-1
    hacer k ← i
    mientras elem(k+1)<elem(k)
        hacer intercambiar (elem(k+1),elem(k))
        hacer k ← k-1
    fin-mientras
siguiente i
```

P4. Se pretende conocer la altura **h** de un pozo midiendo el tiempo que transcurre entre el instante (se supone igual a cero) en el que se lanza una piedra, y el instante **t** en el que el lanzador oye el sonido producido por el choque de la piedra en el agua. Se supone que la velocidad del sonido en el aire es 340 m/s.

Descripción del problema: Suponemos que los t segundos son empleados en su totalidad para la caída de la piedra, en ese caso la profundidad del pozo será: **h** = 1/2 * g * $\mathbf{t}^2$

El valor de **h** así calculado es un valor en exceso, porque **t** es mayor que el valor real del tiempo de caída.

No obstante, podemos utilizarlo, en una primera aproximación para calcular el tiempo $\mathbf{t_{aso}}$ asociado a la propagación de la onda sonora desde el agua al lanzador:

$\mathbf{t_{aso}}$ = **h** / 340. Utilizando este valor podemos recalcular la cota $\mathbf{h_2}$ del pozo de la forma siguiente:

$\mathbf{h_2} = 1/2 * g * (\mathbf{t} - \mathbf{t_{aso}})^2 = 1/2 * g * (\mathbf{t} - \mathbf{h} / 340)^2$, $\mathbf{h_2}$ es ahora menor que la cota real.

Por consiguiente podemos recalcular el tiempo de propagación de la onda sonora, utilizando $\mathbf{h}_2$ en lugar de **h** y obtendremos $\mathbf{t}_{aso} = \mathbf{h}_2 / 340$, y en consecuencia la nueva altura del pozo será

$\mathbf{h_3} = 1/2 * g * (\mathbf{t} - \mathbf{h}_2 / 340)^2$, y así sucesivamente.

Se pide:

a) Escribir un algoritmo en lenguaje natural, que permita calcular la profundidad h de un pozo para un tiempo t (en el que el lanzador oye el sonido producido por el choque de la piedra en el agua), utilizando para ello un subalgoritmo.
El cálculo se interrumpirá cuando se verifique: $| (h_n - h_{n-1}) / h_n | =< 0,001$ **(2,5 puntos)**

```
algoritmo Pozo
variables  t=tiempo medido ; h2=altura temporal
           h=altura pozo , h1=altura asociada
           g=aceleración gravedad
comienzo
      aceptar t
      hacer h1 ← 1/2 * g * t²
      repetir
           hacer h2 ← h1
           hacer h ← 1/2 * g * (t – h1/340)²
           hacer h1 ← h
      hasta abs ((h-h2)/h) <= 0,001
      mostrar h
fin
---------------------------------------
procedimiento abs(n: real)
comienzo
     si n<0 entonces
          hacer n ← -n
     fin-si
fin-procedimiento
```

Universidad de Alicante
Universitat d'Alacant
Escuela Politécnica Superior

COLUMNA	FILA

APLICACIÓN DE ORDENADORES

EXAMEN PRÁCTICO SEPTIEMBRE 2.005 (Duración del examen 2h40m)

APELLIDOS:	Nº CONVOCATORIA:
NOMBRE:	

P1. Se pretende diseñar un algoritmo que genere las calificaciones de los opositores al *Cuerpo de Ingenieros Técnicos de Obras Públicas* obtenidas en el examen tipo test. Para ello, se deben de tener en cuenta las siguientes premisas:

- El test consta de N preguntas
- Cada pregunta del test puede tener K opciones de respuesta
- Por cada K-1 respuestas incorrectas se resta una respuesta correcta
- Las respuestas en blanco no suman ni restan puntos
- Para superar el examen se debe obtener al menos el **60%** de respuestas correctas **netas** (es decir, las respuestas correctas con la reducción provocada por las incorrectas)

Los datos de entrada a considerar son los siguientes:

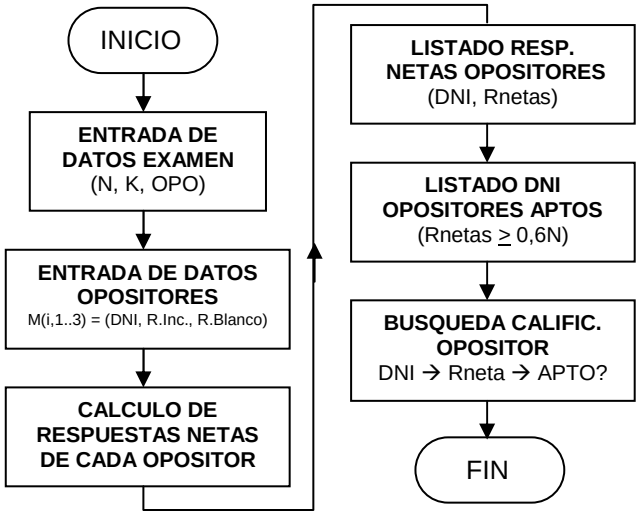
- Número de preguntas del test (N) y opciones de respuesta por pregunta (K)
- Número de opositores presentados
- DNI de cada opositor
- Respuestas incorrectas y en blanco de cada opositor

Se desea obtener:

1.- La relación del número de respuestas netas obtenidas por cada opositor (debe mostrar su DNI y las respuestas netas)
2.- Un listado con los opositores APTOS (debe mostrar únicamente el DNI de los opositores aptos)
3.- Las calificación de cualquier opositor introduciendo su DNI o, si no se encuentra en la relación, la frase "EL DNI NO EXISTE"

Se pide:

a) Realizar un diagrama de bloque que muestre esquemáticamente el proceso a seguir para la resolución del problema planteado. **(1,0 puntos)**



b) Escribir un ÚNICO algoritmo en lenguaje natural que resuelva el citado problema, siguiendo el esquema propuesto en el apartado anterior. **(3,0 puntos)**

```
Algoritmo  Oposicion CITOP
variables      (falta indicar el tipo de variables)
               N = numero de preguntas test
               K = numero de opciones respuesta
               OPO = numero de opositores
               M (i,j) = matriz M de datos de opositores de dimensiones OPO x 4
               (i→opositores, j→datos)
               n, i = contadores
               DNI = DNI del opositor a buscar
               b = condición de parada búsqueda
comienzo
-------------------------------- 1.- Entrada de datos del examen --------------------------------
      aceptar N
      aceptar K
      aceptar OPO

------------------------------ 2.- Entrada de datos de opositores ------------------------------
      para n ← 1 hasta OPO
         aceptar M (n,1)                          (*DNI del opositor*)
         aceptar M (n,2)              (*Respuestas INCORRECTAS del opositor*)
         aceptar M (n,3)                (*Respuestas EN BLANCO del opositor*)
      siguiente n

---------------------- 3.- Calculo de respuestas netas de cada opositor ----------------------
      para n ← 1 hasta OPO
         hacer M (n,4) ← RNETAS ( M(n,2), M(n,3), K-1, N )   (*Cálculo r.netas*)
      siguiente n

---------------------- 4.- Salida de respuestas netas de cada opositor ----------------------
      para n ← 1 hasta OPO
         mostrar "DNI – Rnetas"
         mostrar M(n,1) + "-" + M(n,4)        (*Listado DNI y resp. netas opositores*)
      siguiente n

------------------------------------ 5.- Listado de opositores aptos ----------------------------
      para n ← 1 hasta OPO
         si M(n,4) >= 0,6·N entonces
            mostrar M (n,1)
         fin-si
      siguiente n

--------------------------- 6.- Busqueda calificacion de opositor aptos ----------------------
      aceptar DNI                          (*DNI a buscar en la lista*)
      hacer i ← 0
      mientras b = 0              (*Busqueda secuencial en la lista*)
         hacer i ← i + 1
         si i > OPO entonces b = 1
         si DNI = M (i,1) entonces
            hacer b ← 2
         fin-si
      fin-mientras
      si b = 2 entonces
         mostrar M (i,4)
      si-no
            mostrar "EL DNI NO EXISTE"
      fin-si
fin
```

```
función RNETAS ( i,b,x,t : entero ) : entero
variables  entero: p (contador
             penalización resp)
comienzo
        hacer p ← 0
        mientras i >= x
            hacer i ← i – x
            hacer p ← p + 1
        fin-mientras
        hacer RNETAS ← t – (b + p + i)
fin-función
```

P2. a) Escribir en pseudocódigo un algoritmo que calcule mediante un procedimiento factorial las combinaciones de n elementos tomados de m en m elementos y que viene dado por la expresión:

$$C_n^m = \frac{m\cdot(m-1)\cdot(m-2)\cdot\ldots\cdot(m-n+1)}{n!} = \frac{m!}{n!\cdot(m-n)!}$$

ejemplo : para los números 1,2,3,4, n=2 y m=4 tendremos

$$C_2^4 = \frac{4!}{2!\cdot(4-2)!} = \frac{1\cdot2\cdot3\cdot4}{2!\cdot(4-2)!} = \frac{24}{4} = 6$$

(2 puntos)

```
Algoritmo  Números Combinatorios
variables  (falta indicar el tipo de variables)
           m, n = números a leer
           k = variable auxiliar
           f = factorial
           nu = numerador
           d = denominador
Comienzo
    aceptar m , n
    hacer  k ← m
    factorial ( k , f )
    hacer   un ← f
            k ← n
    factorial ( k , f )
    hacer  d ← f
            k ← m - n
    factorial ( k , f )
    hacer  d ← d * f
    hacer nu ← nu / d
    escribir m + " sobre " + n + " = " + nu
fin
```

```
k = parámetro de entrada   f = parámetro de salida

procedimiento factorial ( a, b: entero )
variables entero: x (contador)
comienzo
        hacer b ← 1
        para x ← 1 hasta a
            b ← b * x

        siguiente x
fin-procedimiento
```

b) Escribir el diagrama de Chapin del apartado (a) **(1 punto)**

Números Combinatorios
comienzo
aceptar m, n
k = m
<<factorial>>
un = f
k = n
<<factorial>>
d = f
k = m- n
<<factorial>>
d = d * f

P3. Escribir en lenguaje natural un algoritmo que permita mostrar por pantalla el triangulo de Tartaglia de la siguiente manera:

(2 puntos)

1	**1**	**1**	**1**	**1**	**1**	**1**
6	**5**	**4**	**3**	**2**	**1**	
15	**10**	**6**	**3**	**1**		
20	**10**	**4**	**1**			
15	**5**	**1**			i - 1, j + 1	
6	**1**			i , j	i , j + 1	
1						

FORMA C (Es necesario que los valores de la matriz estén inicializados a cero 0)

```
Algoritmo Triángulo de Tartaglia
variables: i , j (contadores de filas y columnas respectivamente); n (nº de filas y de columnas de la matriz M); M(i , j) (matriz de
           dimensión n , n); v1 y v2 (suma de valores de elementos de la matriz M); j_ultima (columna última); triangulo
           (acumulador de M(i , j) para la construcción de las filas)
comienzo
        aceptar n
        mientras n >← 1
            hacer i ← 1
                  j ← n
             mientras i <= n
                    si i = 1 entonces
                        hacer v1 ← 0
                        si-no
                            hacer v1 = M(i -1 , j + 1)
                    fin-si
                    si j = n  entonces
                        hacer v2 ← 1
                        si-no
                            hacer v2 ← M(i , j + 1)
                    fin-si
                    hacer M(i , j) ← v1 + v2
                          i ← i + 1
                          j ← j – 1
            fin-mientras
            hacer n ← n - 1
        fin-mientras
        hacer i ← 1
---------------------------------------------------------- visualización ----------------------------------------------------------
        mientras i <= n
            hacer j_ultima ← n – i + 1
                  j ← 1
            mientras j <= j_ultima
                hacer triangulo ←  triangulo + M(i , j) + " "        ' se añade un espacio en blanco
                      j ← j + 1
                mostrar triangulo
            fin- mientras
            hacer triangulo ←  triangulo + "\ n "                    ' se añade un salto de línea
                  i ← i + 1
            mostrar triangulo
        fin-mientras
fin
```

b) Calcular la traza para una matriz 5x5 (n=5)

TRAZA DEL PROBLEMA P3 APARTADO C (FORMA C)

n	i	j	v1	v2	M(i , j)	i	j	n
5	1	5	0	1	1	2	4	
	2	4	1	0	1	3	3	
	3	3	1	0	1	4	2	
	4	2	1	0	1	5	1	
	5	1	1	0	1	6	0	4
4	1	4	0	1	1	2	3	
	2	3	1	1	2	3	2	
	3	2	2	1	3	4	1	
	4	1	3	1	4	5	0	3
3	1	3	0	1	1	2	2	
	2	2	1	2	3	3	1	
	3	1	3	3	6	4	0	2
2	1	2	0	1	1	2	1	
	2	1	1	3	4	3	0	1
1	1	1	0	1	1	2	0	0

	j1	j2	j3	j4	j5
i1	1	1	1	1	1
i2	4	3	2	1	0
i3	6	3	1	0	0
i4	4	1	0	0	0
i5	1	0	0	0	0

n=1 (i,j)	n=2 (i,j)	n=3 (i,j)	n=4 (i,j)	n=5 (i,j)
(1,5)	(1,4)	(1,3)	(1,2)	(1,1)
(2,4)	(2,3)	(2,2)	(2,1)	
(3,3)	(3,2)	(3,1)		
(4,2)	(4,1)			
(5,1)				

Interpretación del contenido:

En fuente normal están redactadas las preguntas del examen y en fuente **negrita** se muestra la solución al examen y las palabras reservadas de la sintaxis algorítmica.

Consideraciones a tener en cuenta al afrontar el examen en esta materia:

- La escritura ha de ser clara, ordenada y legible, aquellos ejercicios que no sean legibles no serán puntuados.
- Ponga especial atención en la sintaxis de los algoritmos.
- Los algoritmos que no sean efectivos (método que, aplicado a resolver un problema, da su solución al cabo de un número finito de etapas y de los correspondientes cálculos) y sin solución tendrán una calificación de 0 puntos.
- El incumplimiento del enunciado del ejercicio dará como resultado la no calificación del ejercicio.
- Si se piensa que un ejercicio no tiene solución hay que demostrarlo sobre el papel razonadamente.