UNIVERSIDAD DE MÁLAGA
E.T.S.I. TELECOMUNICACIÓN

Recursividad

Programación Modular
(1º de Ingeniería de Telecomunicación)

1. Determina qué calcula la siguiente función recursiva. Escribe una función iterativa que realice la misma tarea.

```
ALGORITMO N func(E N n)
VAR
    N sal
INICIO
    SI n == 0 ENTONCES
        sal = 0
    SINO
        sal = n + func(n-1)
    FINSI
    DEVOLVER sal
FIN
```

2. Diseña la versión iterativa del siguiente algoritmo recursivo:

```
ALGORITMO Rec(E N n)
INICIO
    SI NO f(n) ENTONCES
        {cualquier grupo de sentencias que no modifiquen n}
        Rec(g(n))
    FINSI
FIN
```

donde las cabeceras de `f` y `g` se definen como:

```
ALGORITMO B f(E N n)
ALGORITMO N g(E N n)
```

3. Considera la siguiente función recursiva:

```
ALGORITMO Z p(E Z x)
VAR
    Z result
INICIO
    SI x < 3 ENTONCES
        result = x
    SINO
        result = p(x-1) * p(x-3)
    FINSI
    DEVOLVER result
FIN
```

Diseña una función recursiva que calcule el número de productos realizados al ejecutar la función `p` cuando se llama con el argumento `n`.

4. Dada la función recursiva:

```
ALGORITMO Z f(Z x)
VAR
    Z result
INICIO
    SI x>100 ENTONCES
```

```
            result = x-10
        SINO
            result = f(f(x+11))
        FINSI
        DEVOLVER result
    FIN
```

Estudia cuál es su comportamiento. ¿Podrías diseñar `f` de una manera más sencilla?

5. Considera la siguiente función recursiva:

$$Ackerman(m,n) = \begin{cases} n+1 & si\ m = 0 \\ Ackerman(m-1,1) & si\ m > 0 \wedge n = 0 \\ Ackerman(m-1, Ackerman(m,n-1)) & si\ m > 0 \wedge n > 0 \end{cases}$$

Esta función, llamada *función de Ackermann*, es interesante porque crece rápidamente con respecto a los valores `m` y `n`. Comprobar que `Ackermann(1, 2)` vale 4 y que `Ackermann(3, 2)` vale 29. ¿Cuántas llamadas recursivas se hacen cuando queremos evaluar `Ackermann(1,2)`?

6. Para cada uno de los apartados que siguen, implementa una función (o procedimiento, según convenga) recursivo que realice la tarea especificada:

   a) escribir n caracteres '*' en pantalla.

   b) escribir los números del 1 al n.

   c) escribir los números del n al 1.

   d) calcular la suma de dos números naturales utilizando únicamente las funciones `suc` y `pred`, que devuelven, respectivamente el número anterior y siguiente al valor que se le pasa. Ten en cuenta que

$$SumaR(a,b) = \begin{cases} a & si\ b = 0 \\ SumaR(Suc(a), Pred(b)) & si\ b > 0 \end{cases}$$

   e) calcular la potencia de un número usando como operación básica el producto.

   f) escribir las cifras de un número natural.

   g) encontrar el máximo común divisor de dos números siguiendo el algoritmo de Euclides:

$$mcd(a,b) = \begin{cases} mcd(b,a) & si\ a < b \\ mcd(a-b,b) & si\ a > b \\ a & si\ a = b \end{cases}$$

   h) leer una secuencia de caracteres de longitud arbitraria e imprimirla en orden inverso.

7. Para cada uno de los apartados que siguen, implementa una función (o procedimiento, según convenga) recursivo que realice la tarea especificada:

   a) imprima los elementos de un array de longitud n en orden inverso.

   b) sume los n elementos de un array.

   c) evalúe si dos arrays son iguales (tienen los mismos elementos y en el mismo orden).

   d) localice el mínimo y el máximo de un vector de naturales no vacío dado.

   e) implemente el algoritmo de búsqueda secuencial de forma recursiva.

   f) dado un vector de enteros, devuelva un valor booleano que indique si alguno de los elementos del vector coincide con la suma de todos los que le preceden.

8. Implementar soluciones recursivas para ordenar un array siguiendo diferentes métodos:

   a) mediante el método de selección. Se selecciona el mayor de los elementos, se separa y se ordenan los demás.

   b) mediante el método denominado *ordenación por mezcla*, o *Mergesort*:

   - Sea `k` el índice del elemento mitad del array.
   - Ordena recursivamente los elementos hasta `a[k]`, incluyéndolo.

- Ordena recursivamente los elementos siguientes.
- Mezcla de manera ordenada los dos subarrays ordenados en un único array ordenado.

Nota: Para la mezcla ordenada se necesita una array auxiliar.

9. Consideremos los palíndromos sobre un alfabeto formado sólo por letras minúsculas. Sea `P(n)` el número de palíndromos de longitud `n`. Diseña una función recursiva para calcular `P(n)`.
10. Diseña algoritmos recursivos para determinar si una cadena de caracteres que se pasa como parámetro es:
    a) un identificador válido.
    b) un palíndromo.
    c) una palabra perteneciente al lenguaje $L = \{w/w = a^n b^n, n \geq 0\}$.
11. Diseña un algoritmo recursivo que, dado un array de enteros, devuelva el k-ésimo elemento más pequeño del array, es decir, si el array contiene los elementos `(4,7,3,6,8,1,9,2)` y `k=5`, el algoritmo debería devolver `6`.
    Nota: Utiliza el algoritmo `Particion` visto en clase en el método de ordenación `Quicksort`.
12. Desarrollar algoritmos recursivos cuya función sea:
    a) dada una matriz cuadrada, comprobar si es simétrica.
    b) resolver el determinante de una matriz tridiagonal y simétrica. Una matriz tridiagonal simétrica sólo tiene elementos distintos de 0 en la diagonal principal y las dos diagonales adyacentes.

$$\begin{pmatrix} \alpha_1 & \beta_2 & \cdot & \cdot & \cdot & \cdot & 0 \\ \beta_2 & \alpha_2 & \beta_3 & \cdot & \cdot & \cdot & 0 \\ 0 & \beta_3 & \alpha_3 & \beta_4 & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \alpha_{n-1} & \beta_n \\ \cdot & \cdot & \cdot & \cdot & \cdot & \beta_n & \alpha_n \end{pmatrix}$$

El valor del determinante para una matriz de rango r es:

$$det(r) = \begin{cases} \alpha_1 & si\ r = 1 \\ \alpha_1 \times \alpha_2 - \beta_2^2 & si\ r = 2 \\ \alpha_r \times det(r-1) - \beta_r^2 \times det(r-2) & si\ r > 2 \end{cases}$$

13. Diseñar un algoritmo recursivo que devuelva el tamaño de una determinada mancha en un tejido. Una mancha está formada por `'#'` unidos. Dos `'#'` están unidos si son vecinos horizontal, vertical o diagonalmente. Si en el dibujo empezamos en la posición (1, 3), la mancha que incluye esa casilla contiene cinco elementos. Si empezamos en la posición (3, 3), la mancha contiene ocho elementos.

	0	1	2	3	4	5	6	7
0			#	#				#
1		#		#	#		#	
2							#	
3		#		#	#		#	#
4	#					#		

14. Diseñar un algoritmo recursivo que encuentre el camino que ha de seguir un caballo de ajedrez para pasar una sola vez por todas las casillas de un tablero de ajedrez.