



Metodología de Programación E.T.S.I.I. (Gestión y Sistemas)

Relación de problemas

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

```
Algoritmo func(n:N):N
Inicio
    SI n = 0 ENTONCES
        DEVOLVER 0
    ENOTROCASO
        DEVOLVER n+func(n-1)
    FINSI
Fin
```

2.- Diseña la versión iterativa del siguiente algoritmo recursivo:

```
Algoritmo Rec(n:N)
Inicio
    SI NOT f(n) ENTONCES
        {cualquier grupo de sentencias que no modifiquen el valor de n}
        Rec(g(n))
    FINSI
Fin
```

donde las cabeceras de f y g se definen como:

```
Algoritmo f(n:N):B
Algoritmo g(n:N):N
```

3.- Considera la siguiente función recursiva:

```
Algoritmo p(x:Z):Z
Inicio
    SI x < 3 ENTONCES
        DEVOLVER x
    ENOTROCASO
        DEVOLVER p(x-1) * p(x-3)
    FINSI
Fin
```

Sea $A(n)$ el número de productos realizados al ejecutar la función p cuando se llama con el argumento n . Diseña la función recursiva de $A(n)$.

4.- Dada la función recursiva:

```
Algoritmo f(x:Z):Z
Inicio
    SI x>100 ENTONCES
        DEVOLVER x-10
    ENOTROCASO
        DEVOLVER f(f(x+11))
    FINSI
Fin
```

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

5.- Considera la siguiente función recursiva:

$$\text{Acker}(m,n) = \begin{cases} n + 1 & \text{si } m = 0 \\ \text{Acker}(m - 1, 1) & \text{si } n = 0 \\ \text{Acker}(m - 1, \text{Acker}(m, n - 1)) & \text{en otro caso} \end{cases}$$

Esta función, llamada función de "**Ackermann**", es interesante porque crece rápidamente con respecto a los valores m y n . ¿Qué vale $\text{Acker}(1,2)$? ¿Cuántas llamadas recursivas se hacen a la función Acker cuando queremos evaluar $\text{Acker}(1,2)$?

7.- Implementa el algoritmo de búsqueda binaria de forma recursiva.

8.- Escribe un algoritmo recursivo que ordene un array de la siguiente forma:

- Sea k el índice del elemento mitad del array.
- Ordena los elementos hasta $a[k]$, incluyéndolo.
- Ordena los elementos siguientes.
- Mezcla los dos subarrays en un único array ordenado.

Este método de clasificación interna se denomina "**Mergesort**".

9.- 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".

10.- Consideremos palíndromos sobre un alfabeto formado sólo por letras minúsculas. Sea $P(n)$ el número de palíndromos de longitud n . Diseña una función recursiva para $P(n)$.

11.- Diseña un algoritmo recursivo que lea una secuencia de caracteres de longitud arbitraria terminada en un punto, e imprima la suma de todos los dígitos junto con la sucesión en orden inverso de entrada.

12.- Diseña tres algoritmos recursivos para determinar si una cadena de caracteres (que se les pasa como parámetro) es:

- un identificador válido.
- un palíndromo.
- una palabra perteneciente al lenguaje $L = \{w / w = a^n b^n, n \geq 0\}$