Programación Avanzada (ETSINF - UPV)
Examen de ejemplo. Octubre 2023

Nombre: _____

1. 1,5 puntos Escribe el pseudocódigo de una función que devuelva el menor valor de un vector, utilizando una aproximación de divide y vencerás.

Solución:

```
1: function MIN(A, inf, sup)
2:     if inf = sup then
3:         return A[inf]
4:     mid ← ⌊(inf+sup)/2⌋
5:     return min( MIN(A, inf, mid), MIN(A, mid + 1, sup) )
```

2. 3,5 puntos Escribe el pseudocódigo de una función que, dado un vector de enteros, encuentre un pico, es decir, un valor que es mayor que sus vecinos inmediatos a la izquierda y a la derecha. Utiliza la técnica de divide y vencerás.

   Por ejemplo, si

   $$\text{A}=\{5,\ 20,\ \boxed{50},\ 35\}$$

   la función devolverá 50, ya que es mayor que 20 y que 35. Los extremos del vector se consideran un pico si son mayores que su único vecino.

   - Respuesta básica: algoritmo por fuerza bruta que recorre el vector completo (máximo 1 punto)
   - Respuesta completa: solución eficiente usando divide y vencerás (puntuación completa)

Solución:

```
1: function PEAK(A[1..N], inf, sup)
2:     mid ← ⌊(inf+sup)/2⌋
3:     if mid = 1 or mid = N or (A[mid−1] ≤ A[mid] ≥ A[mid+1]) then
4:         return A[mid]
5:     else if A[mid − 1] > A[mid] then
6:         return PEAK(A, inf, mid − 1)
7:     else
8:         return PEAK(A, mid + 1, sup)
```

3. 1.5 puntos Los coeficiente binomiales $C(n,k)$ indican las formas posibles de combinar n elementos tomándolos de k en k. Su valor se puede calcular a través de la expresión

$$C(n,k) = C(n-1,k) + C(n-1,k-1) \quad \text{con } C(n,0) = C(n,n) = 1$$

Este valor se puede calcular mediante la siguiente función recursiva

```
1: function C(n, k)
2:     if k > n then
3:         return 0
4:     if k = 0 or k = n then
5:         return 1
6:     return C(n − 1, k) + C(n − 1, k − 1)
```

Modifica la función para que incluya una función de memoria que evite el recálculo de subsoluciones ya examinadas.

Solución:

```
1: function C(n, k, M)
2:     if M[n][k] ≠ −1 then
3:         return M[n][k]
4:     if k > n then
5:         M[n][k] ← 0
6:     else if k = 0 or k = n then
7:         M[n][k] ← 1
8:     else
9:         M[n][k] ←C(n − 1, k) + C(n − 1, k − 1)
10:    return M[n][k]
```

4. 3.5 puntos Los valores en un vector representan el número máximo de posiciones que se pueden avanzar hacia el final. Queremos saber el número mínimo de saltos que hacen falta para alcanzar (o sobrepasar) el último elemento.

   Por ejemplo, con el vector

   $$A = \{1, 3, 2, 2, 3, 5, 1, 1, 1, 1, 4\}$$

   Resultado: 4 $(1 \to 3 \to 2 \to 5 \to 4)$ Desde 1 solo podemos saltar a 3. Desde el 3, podemos elegir saltar al 2, al otro 2 o al 3. La mejor opción es elegir el segundo 2, que me permite llegar al 5 y desde él al final del vector en un solo salto.

   Escribe el pseudocódigo de una función **que devuelva el mínimo número de saltos** para llegar al final del vector usando la técnica de programación dinámica.

   - Respuesta básica: algoritmo recursivo que examina todos los casos (máximo 2 puntos)
   - Respuesta completa: algoritmo con función de memoria (puntuación completa)

Solución:

```
1:  function JUMP(A, cur, end, M)
2:      if cur = end then
3:          return 0
4:      if M[cur] > 0 then
5:          return M[cur]
6:      njump ← ∞
7:      for i ← 1 to A[cur] do
8:          if cur + i ≤ end then
9:              njump ← mín(njump, 1+ JUMP(A, cur + i, end, M))
10:     M[cur] ← njump
11:     return M[cur]
```