[1 punto] Sea f(n) = $10n^3 + 3$ y se sabe que para $c_1 = 12$ se cumple $10n^3 + 3 \leq c_1 n^3 \ \forall\ n \geq 2$. Y que para $c_2 = 15$ se cumple $10\ n^3 + 3 \leq c_2 n^4 \ \forall\ n \geq 1$. Indica cuáles de las siguientes afirmaciones son ciertas y cuáles son falsas para la función f(n):

a) f(n) es $O(n^4)$
b) f(n) es $O(n^3)$
c) La cota más representativa de f(n) es $O(n^4)$
d) La cota más representativa de f(n) es $O(n^3)$

CIERTAS: a, b, d	**FALSAS: c**

[2 puntos] Completa las siguientes sentencias:

a) $O(n) \subset O(n \log n)$ (*)
b) $\theta(f(n)) + \theta(g(n)) = \theta(\max(f(n), g(n)))$
c) si $f(n) \in \theta(h(n))$ entonces $a\ f(n) + b \in \theta(h(n)) \quad \forall (a \in R^+ \wedge b \in R)$
d) si $t_1(n)$ y $t_2(n)$ son los tiempos de dos implementaciones de un mismo algoritmo, se verifica que $\exists\ c \in \mathbb{R}^+ \wedge \exists\ n_0 \in \mathbb{N} \mid t_1(n) \leq c\ t_2(n)\ \forall\ n \geq n_0$

(*) También era válidas: n^2, n^3, 2^n y n^n

[2.5 puntos] Resuelve la siguiente ecuación de recurrencia a través del método de sustitución, concluyendo con el/los orden/es de complejidad correspondiente/s:

$$T(n) = \begin{cases} c1 & si\ n \leq 2 \\ T(n-1) + T(n-2) + T(ndiv3) + c2 & si\ n > 2 \end{cases}$$

$T(n) = T(n-1)+T(n-2)+T(ndiv3)+c_2 = T(n-2)+T(n-3)+T((n-1)div3) + c_2 +$

$+ T(n-3)+T(n-4)+T((n-2)div3) + c_2 + T(ndiv3-1) + T(ndiv3-2)+T(ndiv3^2)+c_2+c_2=$

…

En la expresión anterior aumenta el número de términos según avanzamos hacia la base, siendo difícil establecer una fórmula general. Para solventar este problema podemos acotar T(n) con las ecuaciones $T_1(n)$ y $T_2(n)$ siguientes:

Acotación superior.-

$$T_1(n) = \begin{cases} c1 & si\ n \leq 2 \\ 3T_1(n-1) + c2 & si\ n > 2 \end{cases}$$

$$T_1(n) = 3T_1(n-1) + c2 = 3[3T_1(n-2) + c2] + c2 = 3^2 T_1(n-2) + 3c2 + c2 =$$

$$= 3^2[3T_1(n-3) + c2] + 3c2 + c2 = 3^3 T_1(n-3) + 3^2 c2 + 3c2 + c2 = \cdots = 3^i T_1(n-i) + \sum_{j=0}^{i-1} 3^j\, c2$$

El caso base se alcanza cuando n-i≤2, por lo que $i \cong n$, de ese modo

$$= 3^n T_1(2) + \sum_{j=0}^{n-1} 3^j\, c2 \in \theta(3^n)$$

Acotación inferior.-

$$T_2(n) = \begin{cases} c1 & si\ n \leq 2 \\ 3T_2(ndiv3) + c2 & si\ n > 2 \end{cases}$$

$$T_2(n) = 3T_2(ndiv3) + c2 = 3[3T_2(ndiv3^2) + c2] + c2 = 3^2T_2(ndiv3^2) + 3c2 + c2 =$$

$$= 3^2[3T_2(ndiv3^3) + c2] + 3c2 + c2 = 3^3T_2(ndiv3^3) + 3^2c2 + 3c2 + c2 = \cdots$$

$$\ldots = 3^iT(ndiv3^i) + \sum_{j=0}^{i-1} 3^j\, c2$$

El caso base se alcanza para $i \cong \log_3 n$, por lo que sustituimos i por su valor en la expresión anterior

$$= 3^{logn}T_2(2) + \sum_{j=0}^{logn-1} 3^j\, c2 \in \theta(n)$$

En conclusión, **$T(n) \in \Omega(n)$ y $T(n) \in O(3^n)$.**

[2.5 puntos] Dada la función Cuestion:

```
Funcion Cuestion ( V[1..num] : vector de enteros; num : entero ) retorna ( s : entero )
var k, p, q: entero fvar

para k = 1 hasta num - 1 hacer
        q = AUX ( V, k, num);
        p = V[ k ]
        V[ k ] = V[ q ]
        V[ q ] = p
fpara
retorna 0
ffuncion

Funcion AUX ( V[1..num] : vector de enteros; inicio, fin: entero) retorna ( t : entero )
var i, m: entero fvar
m = inicio;
para i = inicio + 1 hasta fin hacer
     si ( V[ i ] > V[ m ] )  entonces m = i  fsi
fpara
retorna pos
ffuncion
```

donde la llamada inicial a la función es: **Cuestion(V, num)**
¿Cuál es la talla del problema? ¿Tiene mejor y peor caso?
Calcula su complejidad, mostrando los cálculos necesarios para su obtención.

Funcion AUX.-

Talla del problema: número de componentes de la sección de vector a tratar, es decir, fin – inicio + 1, lo renombraremos como num.

En principio existe mejor y peor caso, debido a la sentencia alternativa si (V[i] > V[m]) ya que en función del cumplimiento de esa condición, o no, se realiza la sentencia m=i.

En el caso de que V[i] <= V[m], sería el mejor caso. En ese supuesto, el coste temporal de la función AUX es el siguiente:

$$T_{MC}(num) = 1 + \sum_{i=inicio+1}^{fin} 1 + 1 = 2 + (fin - inicio - 1 + 1) = 1 + fin - inicio + 1 =$$

$$= 1 + num \in \theta(num)$$

En el caso de que V[i] > V[m], sería el peor caso. En ese supuesto el coste temporal de la función AUX es el siguiente:

$$T_{PC}(num) = 1 + \sum_{i=inicio+1}^{fin} 1 + 1 = 2 + (fin - inicio - 1 + 1) = 1 + fin - inicio + 1 =$$

$$= 1 + num \in \theta(num)$$

En conclusión, ambas complejidades coinciden.

Funcion Cuestión.-

Talla del problema: número de componentes del vector, es decir, num.

$$T(n) = \sum_{k=1}^{num-1} \big(3 + (num - k)\big) = \sum_{k=1}^{num-1} 3 + \sum_{k=1}^{num-1} (num - k) = 3(num - 1) + \sum_{k=1}^{num-1} k =$$

$$= 3num - 3 + \frac{(1 + num - 1)}{2}(num - 1) = 3num + \frac{1}{2}num^2 - \frac{1}{2}num \in \theta(num^2)$$

[2 puntos] Dada la función Cuestion:

```
Q = { 1 ≤ inicio ≤ fin+1 ≤ num + 1 }
Función Cuestion (V[1..num]:vector de enteros; inicio, fin : entero)  retorna ( r : entero )
  var k, p : entero fvar
  p = 1
  si inicio ≥ fin entonces retorna inicio * fin
  si no   k = 1
          mientras k <= 100 hacer
                p = p + V[inicio] * k
                k = k +1
          fmientras
          retorna p * Cuestion (V, inicio + 1, fin – 1 )
  fsi
ffunción
```

donde la llamada inicial a la función es: **Cuestion(V, 1, num)**
¿Cuál es la talla del problema? ¿Tiene mejor y peor caso?
Calcula su complejidad, mostrando los cálculos necesarios para su obtención.

Talla del problema: número de componentes de la sección de vector a tratar, es decir, fin – inicio + 1, lo renombraremos como num.

No existe mejor y peor caso.

La sentencia alternativa (inicio ≥ fin) cuestiona el tamaño del problema, de tal modo que no determina mejor y peor caso. Si inicio ≥ fin, es decir, si la talla del problema es menor o igual que 1, es irrelevante de cara a la eficiencia, pues esta es una propiedad de carácter asintótico.

Para problemas de talla mayor que 1 (inicio < fin), se realiza una llamada recursiva en la que la talla del problema se reduce en dos unidades y previo a la llamada recursiva tiene lugar un bucle *mientras* cuya variable (k) toma un valor inicial de 1 . El número de iteraciones de ese bucle es 100 iteraciones. Siempre se realizan el mismo número de iteraciones, 100, independientemente de la talla, por tanto su complejidad es constante. Dado que en cada iteración del bucle se realizan 2 sentencias de coste unitario, el coste final de esta franja de código es

$$1 + \sum_{i=1}^{100} 2 = 1 + 2(100 - 1 + 1) = 1 + 200 \in \theta(1)$$

Por tanto, la ecuación de recurrencia es

$$T(n) = \begin{cases} c1 & si\ n \leq 1 \\ T(n-2) + c2 & si\ n > 1 \end{cases}$$

expandiendo la recurrencia resulta

$$T(n) = T(n-2) + c2 = T(n-4) + c2 + c2 = T(n-6) + c2 + c2 + c2 =$$

$$= T(n-6) + 3c2 = \cdots = T(n - 2 * i) + ic2$$

El caso base se produce cuando n-2i ≤ 1, por tanto n/2 ≤ i.

Así tenemos que

$$\cong T(0) + \frac{n}{2}c2 = c1 + \frac{n}{2}c2 \in \theta(n)$$