**Examen Parcial 1**

**Nombre y Código**:

| Pregunta | Teorema Maestro | Invariantes | Programación Dinámica | Dividir y Conquistar | Bono | Total |
|---|---|---|---|---|---|---|
| Puntos | 10 | 15 | 10 | 15 | 10 | 60 |
| Puntaje | | | | | | |

**1)** Responda cada una de las siguientes preguntas.

**a)** (2 puntos) Defina que significa que una función *f* pertenezca a *O(f).*

**b)** (3 puntos) Justificando su respuesta, clasifique ascendentemente por orden asintótico las siguientes funciones:

$$f_1(n) = 5^n,\ f_2(n) = n\,log(n),\ f_3(n) = \log(n)\,,\ f_4(n) = n,$$

$$f_5(n) = 4^{\log_2 n},\ f_6(n) = n^3,\ f_7(n) = \sqrt{n},\ f_8(n) = 2^{2n}$$

**c)** (2 puntos) Enuncie el Teorema Maestro.

**d)** (3 puntos) Use el Teorema Maestro para resolver las siguientes recurrencias *T(n)*

i) $T(n) = 2T\left(\frac{n}{2}\right) + n$

ii) $T(n) = 7T\left(\frac{9n}{8}\right) + 1$

iii) $T(n) = T\left(\frac{n}{2}\right) + 1$

**2)** Considere la siguiente especificación:

**Entrada**: un arreglo a[*0..N*) de números, N ≥ 0
**Salida**: el valor del producto de los números en a[*0..N*]

El siguiente fragmento de código Python implementa una solución del problema anterior:

```
1 def prod(a):
2   N = len(a)
3   r,n = 1,0
4   while n != N:
5     r,n = r*a[n],n+1
6   return r
```

**a)** (2 puntos) En términos de la notación *O* y justificando su respuesta, determine la complejidad temporal de la función prod. Puede asumir que las operaciones aritméticas y de asignación toman tiempo *O(1)*.

**b)** (3 puntos) Plantee un invariante para el ciclo while en las líneas 4-5.

**c)** (10) Demuestre las propiedades de iniciación, estabilidad y terminación para su invariante. Concluya que, al momento de terminar, la función *prod(a)* calcula el producto de los números en *a.*

**3)** Dado un arreglo *A[0..N]* de elementos, donde cada elemento tiene un valor y peso asociado a él(*A[0].v* y *A[0].w*), y una mochila con una capacidad máxima W. Deseamos encontrar una selección de elementos que maximicen la suma de los valores de los elementos y que el peso acumulado de todos los ítems elegidos no excedan la capacidad de la mochila.

Considere la siguiente especificación:

**Entrada**: Un arreglo *A[0..N)* de elementos y *W,* con *N ≥ 0 y W ≥ 0*
**Salida**: La máxima suma de valores de elementos que puedo almacenar en la mochila de capacidad W.

Considere una función ø(n,w), con *0 ≤ n ≤ N y 0 ≤ w ≤ W*:

ø(n,w) : "la máxima suma de valores de n elementos que puedo almacenar en una mochila con capacidad w"

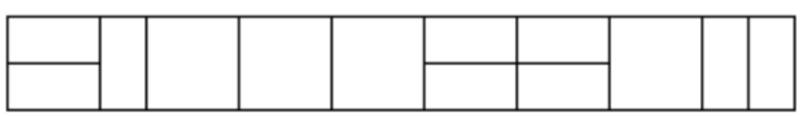
- **a)** (7 puntos) Defina la función recursiva ø(n,w), con todos los posibles casos.
- **b)** (3 puntos) Escriba pseudo-código, para realizar la implementación de la función ø(n,w). Puede utilizar cualquiera de las dos técnicas vistas en clase.

**4)** Considere un arreglo A[0..n), n ≥ *0*, de números enteros. Un segmento creciente de A e un subarreglo A[i, j], con 0 ≤ i < j ≤ n, tal que:

$$A[i] \leq A[i+1] \leq .. \leq A[j-1]$$

- **a)** (2 puntos) De un ejemplo de un arreglo que tenga segmentos crecientes de longitud 3 y 5.
- **b)** (3 puntos) Especifique el problema de terminar la longitud máxima de un segmento creciente de un arreglo.
- **c)** (10 puntos) Usando la técnica de "dividir y conquistar", diseñe un algoritmo que resuelva el problema especificado en el numeral anterior en orden temporal $O(n\ log\ n)$(esto debe demostrarlo)

**5)** Se nos ha consultado determinar de cuantas maneras podemos construir rectángulos de 2*n utilizando bloques con dos formas diferentes. La primera figura es un bloque de 2*1 y la segunda es un bloque de 2*2. A continuación se presenta un ejemplo de cómo hacer un rectángulo de 2*17.



- **a)** (7 puntos) Plantee una función recursiva para calcular la cantidad de formas en que podemos fabricar bloques de tamaño 2*n
- **b)** (3 puntos) Escriba el pseudo-código de su solución para este problema.