APELLIDOS, NOMBRE_____ NOTA_____

Ejercicio_1. (2 puntos)

El algoritmo de ordenación MergeSort puede ser implementado por:

```
MergeSort (A, p, r) /* Ordena un vector A desde p hasta r */
    if p<r { /* Dividir en dos trozos de tamaño igual (o lo más parecido posible), es decir ⌈n/2⌉ y ⌊n/2⌋ */
        q = ⌊(p+r)/2⌋ ;          /* Divide */
        /* Resolver recursivamente los subproblemas */
        MergeSort (A,p,q);       /* Resuelve */
        MergeSort (A,q+1,r) ;    /* Resuelve */
        /* Combinar: mezcla dos listas ordenadas en O(n) */
        Merge (A,p,q,r);         /* Combina*/
      }
```

- El algoritmo utiliza para su implementación la función Merge que tiene un coste $\Theta(n)$.

Se pide:

a. (0,75 puntos). Calcular la complejidad del algoritmo propuesto por el método de la **ecuación característica**.
b. (0,5 puntos). Calcular la complejidad del algoritmo propuesto por el Teorema maestro.
c. (0,75 puntos). Comparar el algoritmo propuesto con el siguiente (Calcular la complejidad y compararlas):

```
MergeSort_bad (A, p, r) /* Ordena un vector A desde p hasta r */
    if p<r {
        MergeSort_bad (A,p,p);
        MergeSort_bad (A,p+1,r) ;
        Merge (A,p,p,r);
      }
```

NOTA: El Teorema maestro es:

$$T(n) \in \begin{cases} O(n^{\log_b a}) & \text{si } a > b^k \\ O(n^k \cdot \log^{p+1} n) & \text{si } a = b^k \\ O(n^k \cdot \log^{p} n) & \text{si } a < b^k \end{cases}$$

➢ **Solución:**

APELLIDOS, NOMBRE_____ NOTA_____

Ejercicio_2. (3 puntos)

Para aprobar una asignatura el/la estudiante tiene que hacer en total **n** tareas (exámenes, prácticas, trabajos, etc). Para cada una de ellas estima que le llevará cierto tiempo, $\mathbf{t_i}$. Como puede realizarlas a lo largo del curso, las quiere repartir entre las convocatorias de junio, septiembre y diciembre. En cada convocatoria puede sacar **M** unidades de tiempo como **máximo**. Se supone que todas las tareas se deben hacer y que no se pueden fraccionar (cada tarea va a una sola convocatoria). El **objetivo** es conseguir un reparto haciendo las tareas cuanto antes, no dejarlas para el final, es decir, minimizar el tiempo dedicado en la convocatoria de diciembre. En caso de empate en diciembre, minimizar el tiempo en la de septiembre.

a. (1 punto). Diseñar un algoritmo **voraz** para resolver el problema aunque no se garantice siempre la solución óptima. Proponer y contrastar dos criterios de selección.
   Aplicar el algoritmo al caso: **n** = 5, **M** = 16, **t** = {7, 5, 3, 5, 6}.
b. (1 punto). Resolver el problema mediante **programación dinámica**. Definir la ecuación recurrente, los casos base, las tablas y el algoritmo para rellenarlas. No hay que aplicar el ejemplo.
c. (1 punto). Resolver el problema por **backtracking** usando el esquema iterativo. Indicar cómo es la representación de la solución, la forma del árbol y programar las funciones genéricas del esquema correspondiente.

NOTA: El esquema iterativo de backtracking puede ser implementado por:

```
Backtracking (var s: TuplaSolución)
    nivel:= 1
    s:= sINICIAL
    fin:= false
    repetir
        Generar (nivel, s)
        si Solución (nivel, s) entonces
            fin:= true
        sino si Criterio (nivel, s) entonces
            nivel:= nivel + 1
        sino mientras NOT MasHermanos (nivel, s) hacer
            Retroceder (nivel, s)
    hasta fin
```

➢ **Solución:**

Ejercicio_3. (1,5 puntos)

Dado el autómata siguiente,

f	a	b
→ 0	1	3
1	0	3
2	1	4
* 3	5	5
4	3	3
* 5	5	3

Obtener:

a. (0.5 puntos). El A.F.D. mínimo equivalente.
b. (0.5 puntos). La expresión regular del lenguaje reconocido por el autómata del apartado anterior.
c. (0.5 puntos). La gramática de tipo 3 para el lenguaje.

➢ **Solución:**

Ejercicio_4. (1,5 puntos)

Dado el lenguaje $\mathbf{(01)^n}$ con $\mathbf{n \geq 0}$,

1) (0,75 puntos). Seleccionar, **justificando la respuesta**. el autómata que reconoce el lenguaje indicado.
   a. AF=[{0,1}, {A,B,C,F}, f, A, {F}] con f(A,0)=B, f(A, λ)= λ, f(C,0)=B, f(B,1)=C, f(B,1)= λ
   b. AF=[{0,1}, {A,B,C,F}, f, A, {F}] con f(A,0)=B, f(A, λ)=F, f(C,0)=B, f(B,1)=C, f(B,1)=F
   c. AF=[{0,1}, {A,B,C,F}, f, A,{F}] con f(A, B)=0, f(A,F)= λ, f(C,B)=0, f(B,C)=1, f(B,F)=1
   d. AF=[{0,1}, {A,B,C,F}, f, A, {F}] con f(B,0)=A, f(F, λ)=A, f(B,0)=C, f(C,1)=B, f(F,1)=B
2) (0,75 puntos). Obtener el AFD mínimo equivalente del autómata seleccionado en el apartado anterior.

➢ **Solución:**

Ejercicio_5. (2 puntos)

Dada la gramática:

$S \rightarrow S = A \mid A$
$A \rightarrow A = B \mid B$
$B \rightarrow (S) \mid a \mid b$

a. (0.5 puntos). Comprobar si es LL(1), eliminar la recursividad a la izquierda y obtener la gramática LL(1) equivalente.
b. (0.5 puntos). Convertir la gramática del apartado anterior en un autómata con pila que acepte el mismo lenguaje por pila vacía.
c. (0.5 puntos). Analizar, teniendo en cuenta el principio de preanálisis (lectura de un símbolo de la entrada con anticipación) la entrada "**(a)**".
d. (0.5 puntos). Implementar el seudocódigo de análisis descendente dirigido por la sintaxis para la gramática obtenida LL(1)

➢ **Solución:**