APELLIDOS, NOMBRE_____ **NOTA**_____

Ejercicio_1. (2 puntos)

Tenemos que ejecutar un conjunto de n tareas, cada una de las cuales requiere un tiempo unitario. En un instante T=1, 2, ... podemos ejecutar únicamente una tarea. La tarea i produce unos beneficios b_i ($b_i > 0$) sólo en el caso en el que sea ejecutada en un instante anterior o igual a d_i.

Diseñar un algoritmo **voraz** para resolver el problema, aunque no se garantice la solución óptima que nos permita seleccionar el conjunto de tareas a realizar de forma que nos aseguremos que tenemos el **mayor beneficio posible**.

- Detallar :

1. (1,5 puntos) Las estructuras y/o variables necesarias para representar la información del problema y el método voraz utilizado (El procedimiento o función que implemente el algoritmo). Es necesario marcar en el seudocódigo propuesto a que corresponde cada parte en el esquema general de un algoritmo voraz. Si hay más de un criterio posible elegir uno razonadamente y discutir los otros. Indicar razonadamente el orden de dicho algoritmo.
2. (0,5 puntos) Aplicar el algoritmo implementado en el apartado anterior a la siguiente instancia:

i	1	2	3	4
b_i	50	10	15	30
d_i	2	1	2	1

Ejercicio_2. (1 punto)

Dado el esquema del algoritmo de ordenación QuickSort:

```
QuickSort (A, izq, der)  /* Ordena un vector A desde izq hasta der */
 if (izq < der) {
  piv=mediana (izq, der)
  div =partition (A, piv,izq, der)
  /*El vector A[izq..der] se particiona en dos subvectores A[izq..div] y A[div+1..der], de forma que los
    elementos de A[izq..div] son menores o iguales que los de A[div+1..der] (según elemento pivote) */
  QuickSort (A, izq, div)
  QuickSort (A, div+1, der)
}
```

Donde, con "***mediana***" se obtiene la mediana de los elementos del array A entre las posiciones izq y der (el elemento que ocuparía la posición central si estuvieran ordenados), y "partition" es el procedimiento de particionar pero usando ***piv*** como pivote, con lo que el problema se divide en dos subproblemas de igual tamaño. Si el tiempo de ejecución del procedimiento "***mediana***"es $t_{med}(n)=20n$, y el de "partition" es $t_{par}(n)=n$:

1. (0.25 pto.) Calcular la complejidad del algoritmo propuesto por el método de la **ecuación característica**.
2. (0.25 pto.) Calcular la complejidad del algoritmo propuesto por el Teorema maestro.
3. (0.25 pto.) Calcular la complejidad del algoritmo propuesto por expansión de recurrencia.
4. (0.25 pto.) Si el método de la **Burbuja** tiene un tiempo de ejecución de n^2, justificar para qué valores de la entrada es preferible esta versión del QuickSort al método de la Burbuja.

NOTAS:

- Teorema: La solución a la ecuación $T(n) = aT(n/b) + \Theta(n^k log^p n)$, con $a \geq 1, b > 1$ y $p \geq 0$, es:

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

- Suma de los valores de la progresión geométrica

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

APELLIDOS, NOMBRE_____ **NOTA**_____

Ejercicio_3. (2 puntos)

- Resolver el problema de la mochila para el caso en que no se permita partir los objetos (es decir, un objeto se coge entero o no se coge nada).
  - ☐ Problema de la mochila.
  - ☐ Tenemos:
    - **n** objetos, cada uno con un peso (p_i) y un valor o beneficio (b_i)
    - Una mochila en la que podemos meter objetos, con una capacidad de peso máximo **M**.
  - ☐ **Objetivo:** llenar la mochila con esos objetos, maximizando la suma de los beneficios (valores) transportados, y respetando la limitación de capacidad máxima **M**.
  - ☐ Se supondrá que los objetos **NO** se pueden partir en trozos.

- **Se pide**:

1. (1 punto) Resolver el problema mediante **programación dinámica**. Definir la ecuación recurrente, los casos base, las tablas y el algoritmo para rellenarlas y especificar cómo se **recompone la solución** final a partir de los valores de las tablas.
   - Aplicar el algoritmo al caso: **n**= 3, **M**= 5, **p**= (1, 1, 4), **b**= (2, 3, 6)
2. (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.
   - Aplicar el algoritmo al caso: **n**= 3, **M**= 5, **p**= (1, 1, 4), **b**= (2, 3, 6)

Ejercicio_4. (2 puntos)

Dado el lenguaje libre de contexto: $\boldsymbol{L = \{a^{2n}b^n c \,/\, n \geq 0\}}$

1. (0.5 pto.) Construir una gramática LL(1) que lo genere.
2. (0.25 pto.) Obtener un *Autómata con Pila* asociado al lenguaje que acepte el mismo lenguaje por pila vacía.
3. (0.25 pto.) Analizar por el autómata anterior, teniendo en cuenta el principio de preanálisis (lectura de un símbolo de la entrada con anticipación) las entradas **"aaaabbc"** y **"abc"**.
4. (0.25 pto.) Con la gramática LL(1), construir la tabla de análisis LL(1) y especificar el pseudocódigo de análisis sintáctico tabular.
5. (0.25 pto.) Construir las trazas correspondientes al reconocimiento de las frases: **"aaaabbc"** y **"abc"** según el pseudocódigo especificado en el punto 4 anterior.
6. (0.5 pto.) Especificar el pseudocódigo de análisis sintáctico LL dirigido por la sintaxis. Analizar si son correctas sintácticamente las entradas: **"aaaabbc"** y **"abc"**

Ejercicio_5. (2 puntos)

Una puerta blindada dispone de una única cerradura. Para abrirla es necesario hacer girar en ella tres llaves diferentes (denominadas a, b y c), en un orden predeterminado, que se describe a continuación:

• Llave a, seguida de llave b, seguida de llave c, o bien

• Llave b, seguida de llave a, seguida de llave c.

Si no se respeta este orden, la caja se bloquea, y es imposible su apertura; por ejemplo, si se hace girar la llave 1, se retira la misma, se introduce de nuevo y se hace girar.

Una vez abierta la caja fuerte, la introducción de las llaves en su cerradura, en cualquier orden, no afecta al mecanismo de cierre (permanece abierta).

Considérese que las denominaciones de las llaves son símbolos de un alfabeto, sobre el que define el lenguaje L cuyas palabras son las secuencias permitidas de apertura de la caja fuerte. Por ejemplo, abcbc es una palabra del referido lenguaje.

Se pide:

a. (0.5 ptos.) Diseño del autómata AFND que acepta el lenguaje L.

b. (0.5 pto.) Gramática que genera las palabras de L.

c. (0.5 pto.) autómata AFD mínimo equivalente del apartado **a.**

d. (0.5 pto.) La expresión regular del lenguaje reconocido por el autómata del apartado c.

APELLIDOS, NOMBRE_____ **NOTA**_____

Ejercicio_6. (1 punto)

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

1. (0.5 pto.) Marcar el autómata que reconoce el lenguaje indicado justificando la respuesta.
   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. (0.5 pto.) Obtener el AFD mínimo equivalente del autómata seleccionado.