Examen. 3 de febrero de 2009

No hay que entregar esta hoja con el examen.
Todas preguntas tienen la misma ponderación en la nota total (25%).
Todas las preguntas deben estar aprobadas por separado.

1. (2,5 puntos) Definimos un tipo de datos **Caja de Naturales** como una colección no ordenada de naturales. Los elementos pueden estar repetidos. La caja tiene una capacidad máxima dada, *M*, de forma que la suma de los elementos contenidos no puede ser mayor que *M*; en otro caso, decimos que la caja se rompe. Construir una especificación formal, usando el método axiomático, del TAD **CajaNaturales**. El tipo tendrá las siguientes operaciones:

   - **crear**: esta operación recibe un número, *M*, y crea una caja vacía con capacidad *M*.
   - **insertar**: dada una caja y un natural, inserta el natural dentro de la caja. Si la suma de los elementos insertados hasta ese momento supera *M*, la caja queda rota.
   - **libre**: dada una caja, devuelve un entero que indica el espacio libre de esa caja, es decir, la capacidad de esa caja menos la suma de los elementos insertados. Se supone que la suma y la resta están ya definidas. Si la caja está rota, el número será negativo.
   - **consulta**: dada una caja y un natural, devuelve un booleano que indica si ese natural ha sido insertado en la caja o no. Si la caja está rota también devuelve falso (se supone que se han *caído* todos sus elementos). Por ejemplo, si se crea una caja de capacidad 18 y se inserta dos veces el 10, entonces consulta de 10 devuelve falso (la caja se ha roto).
   - **arregla**: dada una caja (puede estar rota o no) quita todos los elementos y la deja disponible para insertar más elementos, con la misma capacidad con la que se creó.

   Escribir las cuatro partes de la especificación axiomática (nombre, conjuntos, sintaxis y semántica). Se pueden añadir otras operaciones, que deberán ser especificadas también.

2. (2,5 puntos) En una base de datos de una empresa se almacena información de los empleados. Tenemos un tipo de datos **Persona** y existe una lista global que guarda todas las personas de la base de datos, ordenadas alfabéticamente por el nombre.

   Las consultas en la base de datos pueden ser por nombre o bien por número de DNI. Para hacer rápidas las consultas, decidimos usar tablas de dispersión; en concreto, son tablas de punteros al tipo Persona. Se plantean las siguientes alternativas:
   a) Usar una sola tabla de dispersión abierta, indexada por DNI. Para las consultas por nombre, se hace una búsqueda secuencial en la lista global de personas.
   b) Usar una sola tabla de dispersión abierta, indexada tanto por nombre como por DNI. Es decir, la función de dispersión se puede aplicar sobre cualquier cadena, ya sea un nombre o un DNI. Cada persona se inserta dos veces en la tabla: en la posición h(*nombre*) y en la posición h(*DNI*).
   c) Usar dos tablas de dispersión abierta, una indexada por nombres y otra por DNI. Las dos son independientes, cada una tiene su propia tabla, su función de dispersión, etc.

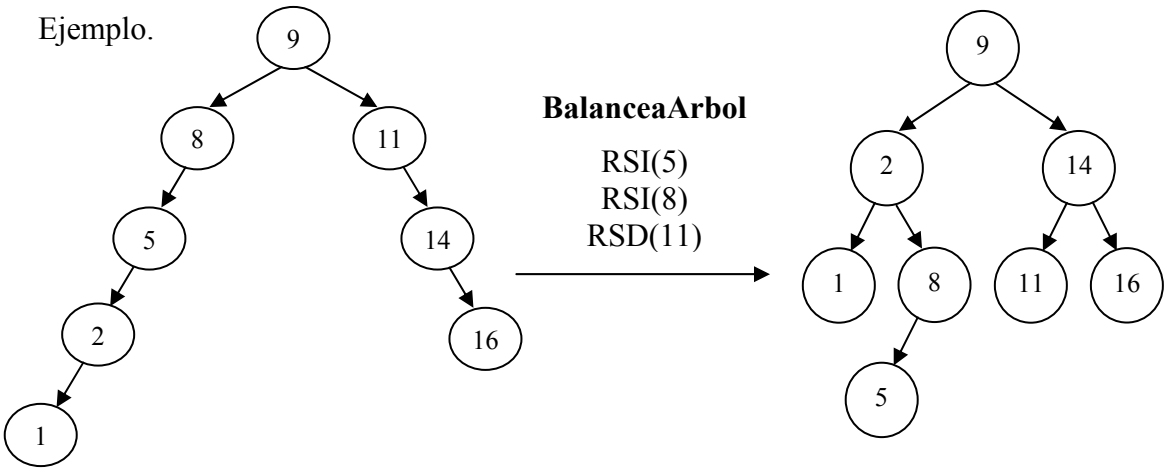
   Calcular los siguientes valores para cada una de las alternativas: (1) uso de memoria; (2) tiempo medio de la inserción de una nueva persona; (3) tiempo medio de una consulta por DNI; (4) tiempo medio de una consulta por nombre. A la vista de los resultados, discutir cuál es más eficiente en cada aspecto.

   Suponer lo siguiente: se almacenan en total *n* empleados, las tablas tienen *B* cubetas, los punteros ocupan 8 bytes y los enteros 4 bytes. No se debe contar la memoria usada en la lista global de personas, por lo que es irrelevante lo que ocupa el tipo de datos Persona.

3. (2,5 puntos) Suponer que tenemos implementada una estructura de árboles binarios de búsqueda. El tipo **Nodo** está compuesto por 4 atributos: **izq** (puntero al hijo izquierdo), **der** (puntero al hijo derecho), **clave** (clave almacenada en el nodo, de tipo T genérico) y **h** (altura de ese subárbol). En este tipo de árboles no existe ninguna comprobación de balanceo, por lo que los árboles resultantes pueden estar desbalanceados.

   Escribir una operación **balanceaArbol** que dado un árbol binario de búsqueda (que puede estar balanceado o desbalanceado), aplique todas las rotaciones necesarias para obtener un árbol balanceado según la condición de los AVL. Es decir, la operación **balanceaArbol** recibe un árbol binario de búsqueda y el resultado es un árbol AVL. No suponer implementada la operación de inserción en un AVL, aunque sí se pueden suponer las operaciones de rotación.

Ejemplo.



4. (2,5 puntos) La ciudad de Cagitán va a instalar una planta de fusión nuclear, con paneles solares en los aparcamientos. Los componentes de la central son muy voluminosos y se requiere el uso de transportes especiales. Estos transportes se caracterizan por que necesitan caminos muy anchos, de varias decenas de metros. Por lo tanto, no nos importa la distancia del camino, sino que sea lo más ancho posible.

   Buscamos el camino más ancho posible entre la fábrica de componentes y la ciudad de Cagitán. Tenemos un mapa de carreteras, con un conjunto de n puntos y m carreteras, donde las carreteras van entre dos puntos. Se supone que la fábrica está en el punto 1 y Cagitán en el n. La matriz A indica en cada posición, $A[i, j]$, la anchura de la carretera de i a j (que es la misma que la que va de j a i) Por ejemplo, si un camino pasa por cuatro carreteras de anchos: 50, 75, 18 y 24, el ancho del camino es 18.

   Escribir un algoritmo que resuelva el problema de manera óptima. Sugerencia: partiendo de algún algoritmo clásico de los vistos en clase, hacer las modificaciones necesarias para que resuelva este problema.

Examen. 3 de febrero de 2009

No hay que entregar esta hoja con el examen.
La primera pregunta cuenta un 30% de la nota de teoría del segundo parcial.
El resto de preguntas tienen la misma ponderación en la nota total (14%).
Todas las preguntas deben ser aprobadas por separado.

1. Resolver la ecuación de recurrencia, con las condiciones iniciales siguientes:

$$T(n) = \sqrt{\frac{1}{2}T^2(n-1) + \frac{1}{2}T^2(n-2) + n} \qquad n > 1$$

$T(0)=1$, $T(1)=1$

2. Tenemos mil cofres cerrados que contienen oro y se encuentran formando una hilera. El cofre más valioso (el cofre del tesoro) está en una posición desconocida. Los cofres que están a la izquierda y a la derecha del cofre del tesoro son menos valiosos cuanto más alejados del mismo. Para saber el valor de cada cofre disponemos de una operación **Pesar**(*i*), que devuelve el valor del cofre *i*. Suponiendo que no existen dos cofres juntos con el mismo peso, es posible encontrar el cofre del tesoro con sólo 20 llamadas a la operación **Pesar**. Se pide:
a) Diseñar un algoritmo divide y vencerás o de reducción que encuentre una forma eficiente de resolver el problema. ¿Garantiza ese algoritmo la solución óptima?
b) Comprobar que con un millón de cofres bastarían 40 pesadas. ¿Cuánto es el tiempo en el caso general?

3. Un constructor dispone de *n* solares y desea construir *n* edificios distintos. El coste de construir cada edificio depende del solar en donde se realice, y viene dado por una tabla. Por ejemplo, en un caso con *n*=4 podríamos tener:

	A	B	C	D
Banco	1	2	8	30
Hotel	4	5	3	12
Colegio	1	7	9	13
Supermercado	7	4	10	18

El constructor desea construir los *n* edificios, cada uno en un solar distinto, pero con el mínimo desembolso económico.
Diseñar un algoritmo voraz que encuentre una buena forma de resolver el problema y determine en qué solar se tiene que construir cada edificio. Hay que ajustarse al esquema y desarrollar sus funciones. ¿Garantiza ese algoritmo la solución óptima?

4. El número de combinaciones de *m* objetos entre un conjunto de *n*, denotado por $\binom{n}{m}$, para $n \geq 1$ y $0 \leq m \leq n$, se puede definir recursivamente por:

$$\binom{n}{m} = 1 \qquad \text{Si } m = 0 \text{ ó } m = n$$

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1} \qquad \text{Si } 0 < m < n$$

Conociendo que el resultado puede ser calculado también con la fórmula n!/(m!·(n-m)!), diseñar un algoritmo de programación dinámica para calcular $\binom{n}{m}$. Describir en detalle la función de recurrencia, los casos base, las tablas, la forma de rellenarlas y la forma de reconstruir la solución. Nota: la tabla construida por el algoritmo es conocida como "el triángulo de Pascal". ¿Cuál será el tiempo de ejecución en este caso?

5. Resolver el problema del ejercicio 3 de forma óptima por backtracking. Se deberán utilizar los esquemas vistos en clase. Definir la forma de representar la solución, el tipo de árbol usado, el esquema y las funciones genéricas del esquema. Seguir los pasos de desarrollo vistos en clase.

6. Resolver el problema del ejercicio 3 de forma óptima por ramificación y poda. Se deberán utilizar los esquemas vistos en clase. Definir la forma de representar la solución, la forma de calcular las cotas y la estimación de beneficio, así como las estrategias de ramificación y poda. Seguir los pasos de desarrollo vistos en clase.