Nombre del alumno:..... Grupo: 3 Tipo A

Sólo se tendrán en cuenta las respuestas que estén en las tablas de respuestas.
Donde se pida uso de memoria, se supondrá que los punteros ocupan 4 bytes y los enteros 4 bytes.

1. Una tabla de dispersión cerrada usa la función de dispersión: $h(x) = (x^2 \text{ div } 10) \bmod B$ (siendo *div* la división entera), y se usa redispersión cuadrática. Se define la siguiente estrategia de restructuración: cuando la tabla esté ocupada más del 73%, se crea otra tabla con el doble de tamaño. Inicialmente el número de cubetas es B = 5. Se insertan los elementos: 7, 3, 4, 9, 2, 10, 8. Mostrar la tabla resultante, indicando de manera gráfica la secuencia de búsqueda de aquellos elementos que producen colisión.

2. Resolver el ejercicio anterior con los mismos datos (igual función de dispersión, B inicial y claves insertadas), pero con dispersión abierta y haciendo restructuración cuando se llene más del 180%.

3. En una aplicación de dispersión, se da el caso de que la tabla está completamente llena. Tenemos la opción de usar dispersión abierta, o dispersión cerrada que puede ser con punteros o sin punteros (una tabla con los elementos). Ordenar las tres opciones de menor a mayor uso de memoria para ese caso concreto. (El resultado es independiente de los datos que se almacenen.)

4. Para representar conjuntos de números enteros de 32 bits tenemos las tres siguientes opciones: arrays de booleanos, listas ordenadas de elementos, y tablas de dispersión cerrada. Indicar qué estructura es más eficiente (en término promedio) para cada una de las siguientes operaciones: (a) Listar los elementos del conjunto de menor a mayor; (b) eliminar un elemento; (c) hacer la intersección de dos conjuntos.

5. En una tabla de dispersión cerrada se utiliza la función de dispersión h(x) = x mod B, siendo B = 10. La función de redispersión es de la forma: $h_i(x) = (h(x) - 3 \cdot i) \bmod B$. Se elimina el elemento 10 y luego el 36. Se quiere hacer sin usar marcas de eliminado. Mostrar la tabla resultante después de cada una de las eliminaciones.

0	1	2	3	4	5	6	7	8	9
80	41		26	10		36	27	17	

6. Tenemos una aplicación de tablas de dispersión abiertas para almacenar datos de personas, indexadas por el DNI. Se han propuesto las siguientes funciones de dispersión. Ordenarlas desde la más buena hasta la más mala. Se supone que M es el DNI de la persona y B es 1000.
   a) h(M) = M mod B b) h(M) = (M div 1000 + M) mod B c) $h(M) = (M \text{ div } 10^6) \bmod B$
   d) h(M) = random() mod B (donde random() es un generador de números aleatorios de 32 bits)

7. Indicar la afirmación o afirmaciones que son ciertas sobre las tablas de dispersión:
   a) En promedio, con igual número de cubetas, la dispersión abierta es más eficiente que la cerrada.
   b) En el peor caso de la función de dispersión, es mejor usar dispersión cerrada con una redispersión adecuada, por ejemplo redispersión doble.
   c) En dispersión abierta, las estrategias de reestructuración de las tablas hacen que las operaciones sean más ineficientes, al tener que recolocar todos los elementos en la tabla nueva.
   d) Una función de dispersión muy costosa de calcular puede hacer que se pierda la eficiencia de las tablas de dispersión.

8. En cierta aplicación de conjuntos, las claves almacenadas son ángulos que vienen dados en radianes (entre $-\pi$ y π). Definir una función de dispersión que pueda ser adecuada para este caso, para tamaños de la tabla suficientemente grandes.

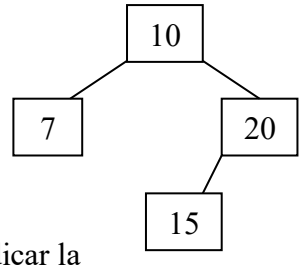
9. Calcular cuándo se debe reestructurar una tabla de dispersión cerrada (es decir, el porcentaje de llenado n/B) para conseguir que las secuencias de búsqueda no sobrepasen (en promedio) de tamaño 5.

10. Una tabla de dispersión cerrada con enteros utiliza la función de dispersión h(x) = x mod B, y redispersión doble: $h_i(x) = (h(x) + i*C(x)) \bmod B$. Suponer que la tabla tiene inicialmente tamaño B=2, y cuando se llena se duplica el número de cubetas. Definir una función C(x) adecuada para este caso.

Tema 2

Respuestas:

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	



1. Sobre el árbol AVL mostrado a la derecha, indicar un elemento que al insertarlo en el árbol produzca:
   a) Una rotación simple a la derecha.
   b) Una rotación doble a la derecha.
   c) Una rotación simple a la izquierda.
   d) Una rotación doble a la izquierda.

   Si no hay ninguno que lo pueda producir, habrá que indicarlo. Y si hay alguno, indicar la rotación que produce. Por ejemplo: "Si se inserta X, se produce una rotación… sobre el nodo…".
2. Indicar cuál o cuáles de las siguientes afirmaciones sobre las operaciones de rotación son ciertas:
   a) Cualquier rotación aplicada sobre un árbol binario de búsqueda produce un árbol binario de búsqueda (suponiendo que la rotación pueda ser aplicada).
   b) Cualquier rotación aplicada sobre un AVL produce un AVL.
   c) Cualquier rotación aplicada sobre un árbol perfectamente balanceado produce un árbol perfectamente balanceado.
   d) La respuesta c) solo es cierta cuando se trata de rotaciones simples.
3. Suponer la operación de eliminación de un elemento de un árbol AVL. Considerar que estamos en un nodo **A**, que se ha hecho la eliminación recursiva por su rama derecha, y como consecuencia de ello el nodo **A** ha quedado desbalanceado. Implementar (en cuatro líneas como máximo) la comprobación que se hace y la rotación que se hace en cada caso. Suponer que en un nodo **X** podemos acceder a: X.der, X.izq, X.clave, altura(X), RSI(X), RSD(X), RDD(X) y RDI(X).
4. En una estructura de relaciones de equivalencia, con balanceo de árboles y compresión de caminos, de tamaño 10 aplicamos las siguientes operaciones. Mostrar la tabla resultante y representar de forma gráfica los árboles obtenidos. Operaciones: Unión(4, 3), Unión(3, 7), Unión(2,6), Unión(3, 5), Unión(8, 1), Unión(3, 6), Unión(6, 1).
5. Sobre el resultado del ejercicio anterior, indicar cuántas clases de equivalencia existen al final y qué elementos contiene cada clase.
6. Como sabes, los nodos de los árboles trie se pueden representar con arrays o con listas. Queremos introducir la novedad de que en un mismo árbol algunos nodos se representen con arrays y otros con listas, según tengan más o menos hijos. En concreto, si un nodo tiene pocos hijos, se representará con listas, y si tiene muchos con arrays. Calcular a partir de qué número de hijos se debe utilizar una u otra representación, para un uso de memoria óptimo. Suponer que cada carácter se almacena en 1 byte y que el tamaño del alfabeto (sin incluir la marca de fin "$") es 30.
7. En un árbol B de orden p = 4 inicialmente vacío, se insertan los elementos: 42, 73, 15, 81, 50, 90, 30, 8, 1. Mostrar el árbol B resultante.
8. Sobre el árbol B resultante del ejercicio anterior, indicar dos elementos que al insertarlos hagan que la altura del árbol aumente. Mostrar el árbol resultante de dichas inserciones.
9. Un sistema operativo almacena la estructura de directorios y archivos de un disco utilizando árboles B. Cada nodo del árbol se almacena en un sector del disco. Al estar la información en disco, no se usan punteros sino direcciones en disco, cada una de las cuales ocupa 8 bytes. Además, la información de cada archivo o directorio ocupa 128 bytes. Suponiendo que los sectores del disco son de 32 Kbytes, calcular el orden p óptimo del árbol B.
10. El siguiente algoritmo recursivo escribe todas las palabras de un trie en orden alfabético, siendo la llamada inicial: **Listar(raiz, "")**.

```
operación Listar (t: trie; p: cadena)
        para cada carácter c hijo del nodo t hacer
                si c==$ entonces Escribir(p)
                sino Listar (Consulta(t, c), p+c)        // p+c es una concatenación
        finpara
```

    Modificar esta operación para que encuentre todas las palabras del trie que solo contengan las letras "A", "B" o "C" (por ejemplo, "BACA"; "BABA", "ACABA"). Solo hay que indicar la parte que cambia de la operación.

Tema 3

Respuestas:

1	
2	
3	
4	
5	
6	
7	
8	
9	
10	