Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED diciembre 2008
Modalidad 0

Normas:
- La entrega del test no corre convocatoria.
- Tiempo para efectuar el test: **20 minutos**.
- Una pregunta mal contestada elimina una correcta.
- Las soluciones al examen se dejarán en el campus virtual.
- **Una vez empezado el examen no se puede salir del aula hasta finalizarlo.**
- En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F		
Los enriquecimientos forman parte de la definición de un TAD.	❑	❑	1	F
Sea el método *Primera* perteneciente a la clase *TLista* que devuelve la primera posición de la lista que lo invoca: *TPosicion TLista::Primera( )* *{ TPosicion p;* *p.pos = lis;* *return p; }* *class TLista {* *public: ...* *private:* *TNodo *lis; }* En el método Primera, se invoca a la sobrecarga del operador asignación entre objetos del tipo *TPosicion*.	❑	❑	2	F
En C++, las funciones y clases AMIGAS es obligatorio declararlas siempre antes de la sección PUBLIC de la clase	❑	❑	3	F
En la escala de complejidades se cumple que O(log n) ⊂ O(log log n).	❑	❑	4	F
Sea el tipo *cola* definido en clase. La semántica de la operación *cabeza* es la siguiente: *Var c:cola; x:item;* *cabeza(crear_cola())=error_item()* *si esvacia(c) entonces cabeza(encolar(c,x))=x* *sino cabeza(encolar(c,x)))=encolar(cabeza(c),x)*	❑	❑	5	F
El nivel de un nodo en un árbol coincide con la longitud del camino desde la raíz a dicho nodo	❑	❑	6	F
A los árboles generales también se les llama árboles multicamino de búsqueda	❑	❑	7	F
Un árbol completo siempre está balanceado respecto a la altura	❑	❑	8	V
El número mínimo de elementos que se pueden almacenar en un árbol 2-3 de altura *h* es 2^h-1	❑	❑	9	V
Para que decrezca la altura de un árbol 2-3-4 en una operación de borrado, el nodo raíz y sus hijos tienen que ser 2-nodo	❑	❑	10	V
Un árbol rojo-negro es un árbol binario balanceado respecto a la altura	❑	❑	11	F
El árbol 2-3 es un árbol B m-camino de búsqueda con m=2	❑	❑	12	F
Sea una tabla de dispersión cerrada con estrategia de redispersión $h_i(x)=(H(x) + C*i)$ MOD B, con B=1000 y C=74. Para cualquier clave "x" se recorrerán todas las posiciones de la tabla buscando una posición libre.	❑	❑	13	F
Para todo nodo de un árbol Leftist, se cumple que el número de nodos de su hijo izquierdo es menor que el de su hijo derecho.	❑	❑	14	F
En un multigrafo pueden existir infinitas aristas para un número "n" de vértices.	❑	❑	15	V

Examen PED diciembre 2008

Normas:
- Tiempo para efectuar el ejercicio: **2 horas**
- En la cabecera de cada hoja **Y EN ESTE ORDEN** hay que poner: ***APELLIDOS, NOMBRE***.
- Cada pregunta se escribirá en hojas diferentes.
- Se dispone de 20 minutos para abandonar el examen sin que corra convocatoria.
- Las soluciones al examen se dejarán en el campus virtual.
- Se puede escribir el examen con lápiz, siempre que sea legible
- **Todas las preguntas tienen el mismo valor.** Este examen vale el 60% de la nota de teoría.
- **Publicación notas**: se publicará un anuncio en el campus virtual.
- **Los alumnos que estén en 5ª o 6ª convocatoria deben indicarlo en la cabecera de todas las hojas**

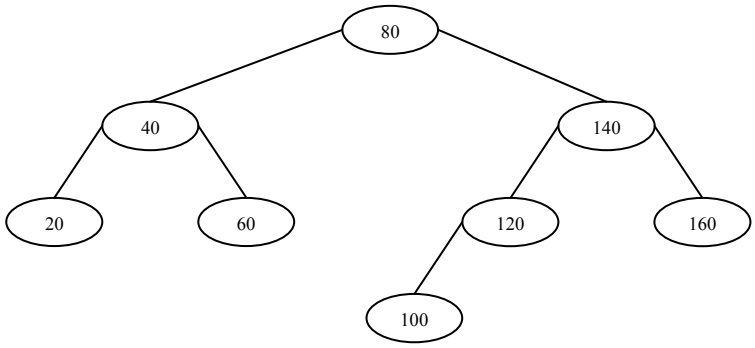
1. Sea un árbol binario cuyas etiquetas son números naturales. Especificar la sintaxis y la semántica de la operación *arbLleno*, que comprueba si el árbol binario cumple las propiedades de árbol lleno.

2. Dado el siguiente árbol AVL, realizar las siguientes operaciones en el siguiente orden:

a) Insertar las etiquetas 90, 170, 220, 10, 2
b) Borrar las etiquetas: 120, 80

Indicar en cada caso las transformaciones realizadas. Emplear los siguientes criterios: en caso de un nodo interior, sustituir por el mayor de la izquierda.

NOTA: Si el árbol que se obtiene en uno de los apartados no es el correcto, no se corregirán los siguientes apartados.



3. Sea un montículo doble (DEAP) de enteros inicialmente vacio:

a) Insertar los siguientes enteros 7, 9, 12, 8, 6, 15, 22, 31, 5, 4, 2, 1.
b) Del montículo doble resultante del apartado *a)*, extraer los 3 enteros de menor prioridad.

4. Dar **razonadamente** las complejidades temporales en el peor y mejor caso de las siguientes operaciones (se exigirá que la justificación de la complejidad sea correcta):

a) Inserción en un árbol 2-3.
b) Búsqueda binaria de un elemento en un vector ordenado.
c) Inserción de un elemento en una lista desordenada que no permite elementos repetidos.

Examen PED diciembre 2008. Soluciones

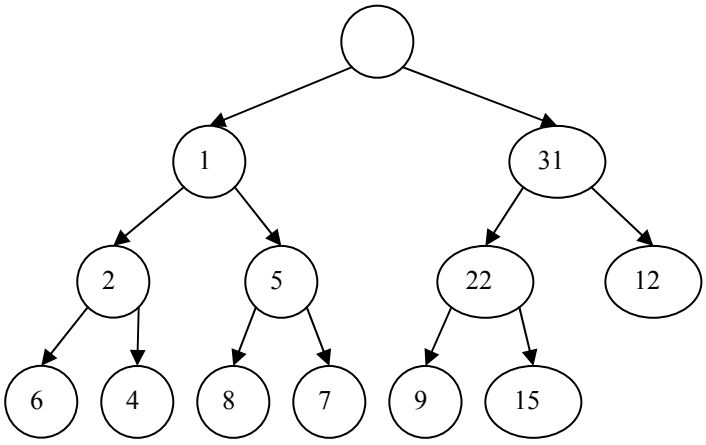
1.

```
MODULO ARBOLES BINARIOS
    USA BOOL, NATURAL

    PARAMETRO TIPO item
        OPERACIONES
            <,==,>:item,item → bool
            error_item() → item
    FPARAMETRO
    TIPO arbin
    OPERACIONES
    crea_arbin() → arbin
    enraizar(arbin, item, arbin) → arbin
    raiz(arbin) → item
    esvacio(arbin) → bool
    hijoiz,hijode(arbin) → arbin
    altura(arbin) → natural
    arbLleno(arbin) → bool
    arbCompleto(arbin) → bool

    VAR i,d:arbin; x,y:item

    ECUACIONES
    arbLleno(crea_arbin()) = TRUE
    arbLleno(enraizar(crea_arbin(),x,crea_arbin()) = TRUE
    arbLleno(enraizar(crea_arbin(),x,enraizar(i,y,d)) = FALSE
    arbLleno(enraizar(enraizar(i,y,d),x,crea_arbin()) = FALSE
    arbLleno(enraizar(i,x,d)) =
        si (altura(i) == altura(d)) entonces
            si (arbLleno(i) y arbLleno(d)) entonces
                TRUE
            sino
                FALSE
            fsi
        sino
            FALSE
        fsi
```

2.

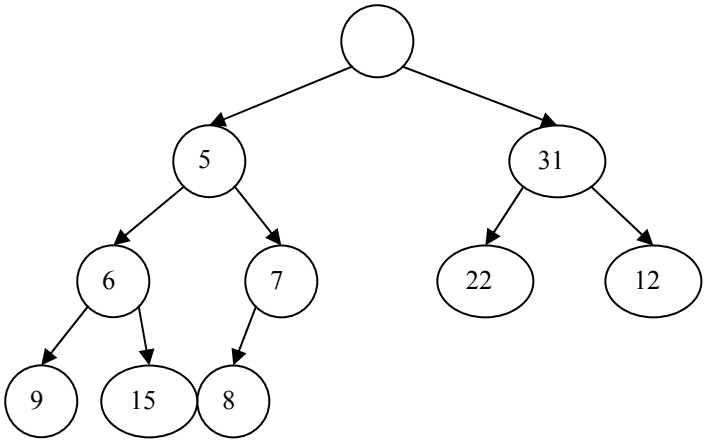
3.

Apartado a)



SOLUCION:

Apartado b)



4.

a) $O(\log_2(n))$ con *n* el número de elementos, ya que se ha de recorrer el árbol como máximo desde la raíz hasta las hojas y nuevamente hasta la raíz, con lo que la complejidad queda en función de la altura que será máxima en su caso peor cuando todos los nodos del árbol sean 2-nodo. $\Omega(\log_3(n))$, con la misma explicación anterior, salvo que sólo se realizará el recorrido descendente y la altura será mínima cuando todos los nodos sean 3-nodo.

b) $O(\log_2(n))$ cuando encuentre el elemento en el último intento, al ir dividiendo sucesivamente el espacio de búsqueda por dos, con *n* la dimensión (o tamaño) del vector; y $\Omega(1)$ cuando lo encuentre en el primer intento.

c) $O(n)$ y $\Omega(1)$, con *n* el número de nodos de la lista, ya que siempre antes de la inserción habrá que recorrer toda la lista para comprobar que el elemento no estuviese previamente en ella. El peor caso será cuando se inserte el elemento o bien que encuentre el elemento repetido en la última posición de la lista; y el mejor caso cuando encuentre el elemento repetido en la primera posición de la lista.