Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED diciembre 2010
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		
Sea el método Primera perteneciente a la clase TLista que devuelve la primera posición de la lista que lo invoca: *TPosicion TLista::Primera( )* *class TLista {* *{ TPosicion p;* *public: ...* ***p.pos = lis;*** *private:* *return p; }* *TNodo *lis; }* En la línea resaltada, se invoca a la sobrecarga del operador asignación entre objetos del tipo TPosicion.	☐	■	1	F
En la escala de complejidades se cumple que $O(n \log n) \subset O(n^2)$.	■	☐	2	V
La operación BorrarItem tiene la siguiente sintaxis y semántica: BorrarItem: LISTA, ITEM -> LISTA BorrarItem(Crear, i) = Crear BorrarItem(IC(L1,j), i) = si (i == j) entonces L1 sino IC (BorrarItem (L1, i), j) Esta operación borra la primera ocurrencia del item que se encuentra en la lista	■	☐	3	V
El nivel de un nodo en un árbol coincide con la longitud del camino desde la raíz a dicho nodo	☐	■	4	F
A los árboles generales también se les llama árboles multicamino de búsqueda	☐	■	5	F
Cuando se realiza una inserción en un AVL, en el camino de vuelta atrás para actualizar los factores de equilibrio, sólo se va a efectuar una rotación como mucho	■	☐	6	V
Dado un árbol 2-3 con *n* items con todos sus nodos del tipo 2-Nodo: la complejidad de la operación de búsqueda de un ítem es $O(\log_2 n)$	■	☐	7	V
Un árbol Rojo-Negro es una representación sobre árbol binario de un árbol 2-3	☐	☐	8	F
Un árbol rojo-negro es un árbol binario balanceado respecto a la altura	☐	☐	9	F
La raíz del árbol B m-camino de búsqueda siempre tiene al menos m/2 claves o etiquetas	☐	☐	10	F
La especificación algebraica de la siguiente operación eliminaría todas las claves repetidas de un determinado ítem (C: ConjuntoConClavesRepetidas; x, y: Ítem): *Eliminar(Crear, x) ⇔ Crear* *Eliminar(Insertar(C, x), y) ⇔* *si (x == y) entonces C sino Insertar(Eliminar(C, y), x)*	☐	■	11	F
En una tabla de dispersión cerrada con la siguiente función de redispersión para la clave 14: $h_i(14)=(28 + 7*i) \text{ MOD } 2000$, se recorrerán todas las posiciones de la tabla buscando una posición libre.	■	☐	12	V
El TAD Cola de Prioridad representado por un montículo, tendrá las siguientes complejidades: $O(1)$ para el borrado, y $O(\log n)$ para la inserción, siendo *n* el número de elementos.	☐	■	13	F
En un árbol binario lleno, el camino mínimo de la raíz (longitud del camino más corto hasta un árbol vacío) es igual a la altura del árbol.	■	☐	14	V
La altura máxima de un árbol de búsqueda digital es "n+1", siendo n el número de bits de la clave.	☐	☐	15	V

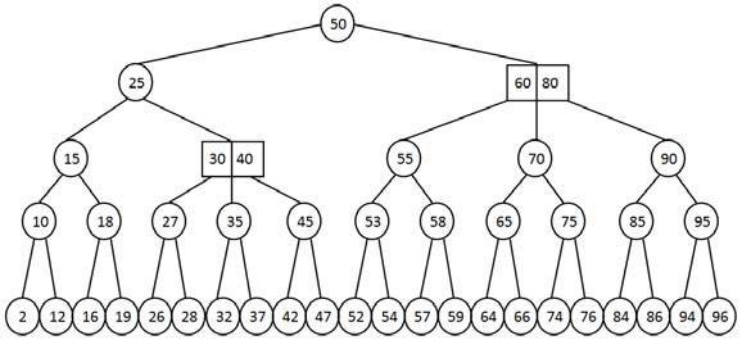
Examen PED diciembre 2010

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.
- **Los alumnos que estén en 5ª o 6ª convocatoria deben indicarlo en la cabecera de todas las hojas**

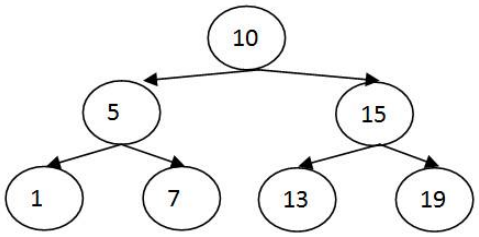
1. Realizar la especificación algebraica de una función que determine si un árbol binario cumple las condiciones de balanceo de un AVL. Para ello, se supone que están definidas las siguientes operaciones:

- Resta de naturales: *resta(a,b)*, *a* debe ser mayor o igual que *b*.
- Comparación de naturales: <=, >=, ==, >, <
- Operaciones booleanas: AND, OR.
- Y todas las operaciones del tipo de datos árbol binario.

2. Borrar en el siguiente árbol 2-3 los elementos: 50, 2 y 80. Criterios: (1) si el nodo tiene dos hijos hay que sustituir por el mayor de la izquierda, (2) si el 2-nodo tiene dos hermanos, consultar el hermano de la izquierda. Nota: en el borrado de cada elemento, indicar las operaciones (transformaciones) realizadas:



3. Sea el siguiente árbol AVL:



a) Insertar los ítems 1,2,4,8,11,12.
b) ¿El árbol resultante del apartado *a)* es un leftist? Explica porqué.
c) Si consideras que no es un leftist, realiza los mínimos cambios imprescindibles para convertir el árbol resultante del apartado a en un leftist. Explica las acciones que tomes.

4. Dar **razonadamente** las complejidades temporales en el peor y mejor caso de las siguientes operaciones en función del parámetro *n*. Se exigirá que la justificación de la complejidad sea correcta:

a) Inserción en un árbol B con *m=5* y *n* elementos.
b) Búsqueda de un elemento en una lista ordenada de *n* elementos.
c) Borrado de un elemento en un montículo con *n* elementos.

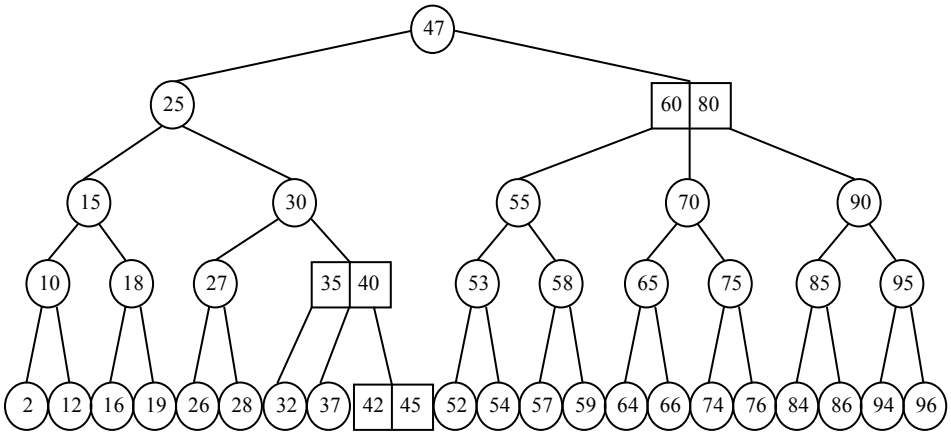
Examen PED diciembre 2010. Soluciones

1.

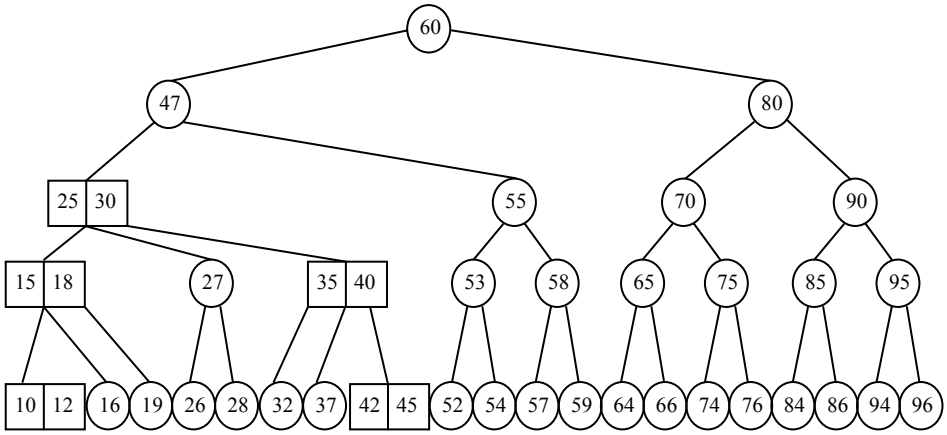
```
VAR i,d: arbin; x: item

ES_AVL(arbin) → bool

ES_AVL(crear()) = V
ES_AVL(enraizar(crear_arbin(), x, crear_arbin())=V

ES_AVL(enraizar(i, x, d) ) =

si altura(i) >= altura(d)
entones
        si resta(altura(i), altura(d)) >= suc(suc(cero))
        entones
                F
        sino
                ES_AVL(i) y ES_AVL(d)
        fsi
sino
        si resta(altura(d), altura(i))>=suc(suc(cero))
        entones
                F
        sino
                ES_AVL(i) y ES_AVL(d)

        fsi
fsi
```

2.

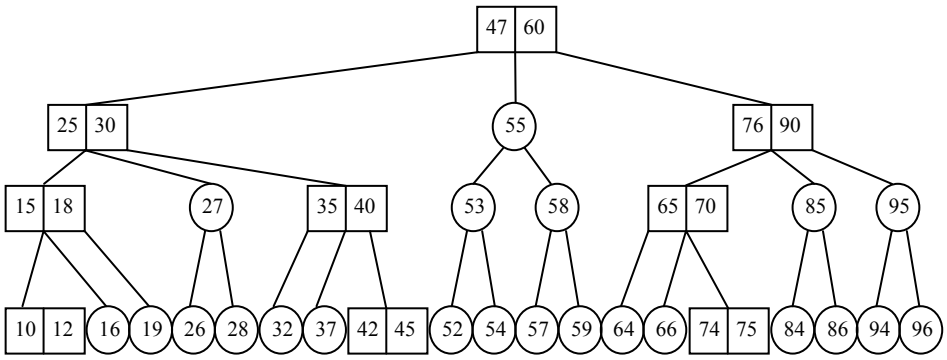
Borrado del 50 → 2 combinaciones



Borrado del 2 → 3 combinaciones y 1 rotación

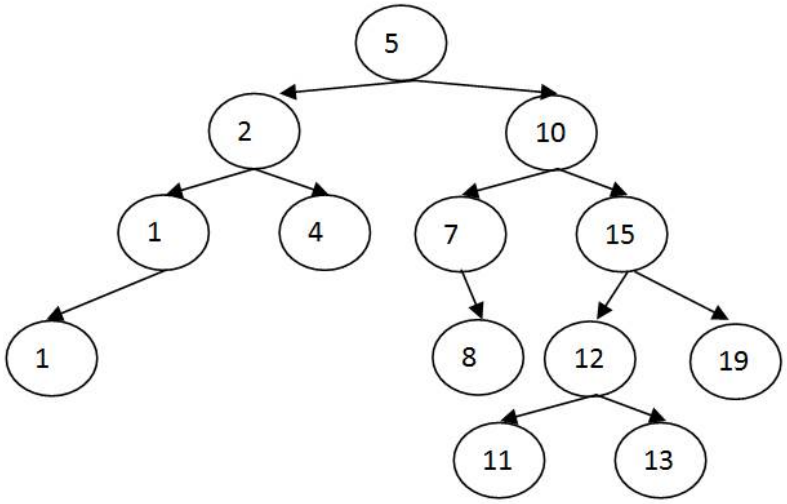


Borrado del 80 → 4 combinaciones



3.

a) Hay que tener en cuenta que el 1 ya existe. Según la propiedad de camino de búsqueda, el ítem del hijo izquierda tiene que ser menor o igual que el ítem de la raíz y la raíz menor o igual que el ítem del hijo derecha. Es por ello que en el caso de repetidos, la búsqueda de un repetido sigue por la izquierda y no por la derecha.



b) No es leftist, porque el nodo 7 no cumple la condición de leftist ni el nodo 10.
c) Para convertirlo en leftist con las menos acciones posibles, hay que intercambiar los hijos del nodo 7 y los hijos del nodo 10.

4.

a) $O(\log_3(n))$ 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 3-nodo. $\Omega(\log_5(n))$, con la misma explicación anterior, salvo que sólo se realizará el recorrido descendente y la altura será mínima cuando todos los nodos estén llenos (5-nodo).

b) O(n), cuando el elemento a buscar esté en la última posición de la lista; y $\Omega(1)$ cuando lo encuentre en el primer intento.

c) $O(\log_2(n))$ y $\Omega(\log_2(n))$, ya que siempre habrá que recorrer un camino descendente desde la raíz hasta las hojas del montículo.