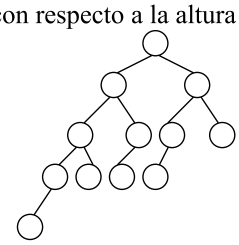
Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED julio 2011
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		
En C++, la expresión return &c; devuelve la dirección de memoria de la variable c.	■	□	1	V
En C++, una función no puede tener todos sus parámetros con valores por omisión o por defecto.	□	■	2	F
En la escala de complejidades se cumple que $O(\log n) \subset O(\log \log n)$.	□	■	3	F
La operación BorrarItem, que borra todas las ocurrencias del item i que se encuentren en la lista, tiene la siguiente sintaxis y semántica: BorrarItem: LISTA, ITEM -> LISTA BorrarItem(Crear, i) = Crear BorrarItem(IC(L1,j), i) = si (i == j) entonces BorrarItem (L1, i) sino IC (BorrarItem (L1, i), j)	■	□	4	V
Un árbol con un único nodo tiene un único camino cuya longitud es 1	□	■	5	F
En cualquier tipo de datos árbol, cada elemento puede tener varios predecesores, pero como máximo un sucesor.	□	■	6	F
El siguiente árbol está balanceado con respecto a la altura 	■	□	7	V
Si se inserta un elemento en un árbol 2-3 y todos los nodos que están en el camino desde la raíz a la hoja donde se inserta el elemento son del tipo 3-nodo, la altura del árbol 2-3 resultante crece con respecto al árbol 2-3 original.	■	□	8	V
En un árbol 2-3-4 de altura 3 donde todos sus nodos son del tipo 3-nodo, el número de elementos total es 27.	□	■	9	F
En un árbol rojo-negro, el número de enlaces negros ha de ser mayor que el de enlaces rojos.	□	□	10	F
El nodo de un árbol B m-camino de búsqueda con m=100 puede tener como máximo 99 claves.	□	□	11	V
La complejidad temporal, en su peor caso, de la operación de Unión entre 2 conjuntos con m elementos cada uno y representados con una lista ordenada es 0(m).	■	□	12	V
En el Hash cerrado la tabla de dispersión de tamaño B se tiene que reestructurar cuando se cumpla que el número de elementos $n \geq 2B$.	□	■	13	F
En un TRIE la complejidad temporal en su peor caso de la función Pertenece es O(n) siendo n el número de nodos del árbol	□	□	14	F

Examen PED julio 2011

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

1. Dar la sintaxis y la semántica de la operación **mezcla**, que actúa sobre dos colas y devuelve una cola nueva con los elementos de las dos colas encolados de forma alternada y en el mismo orden que aparecen en cada cola.

Por ejemplo, suponiendo que C1 y C2 son dos colas, escritas de izquierda a derecha desde la cabeza al fondo de la cola, la mezcla de C1 y C2 debería dar el siguiente resultado:

C1 = (a b c)
C2 = (x y)

mezcla(C1, C2) = (a x b y c)

2. Dados los siguientes recorridos de un árbol:

Preorden: 16, 12, 30, 21, 4, 8
Inorden: 12, 30, 16, 4, 8, 21

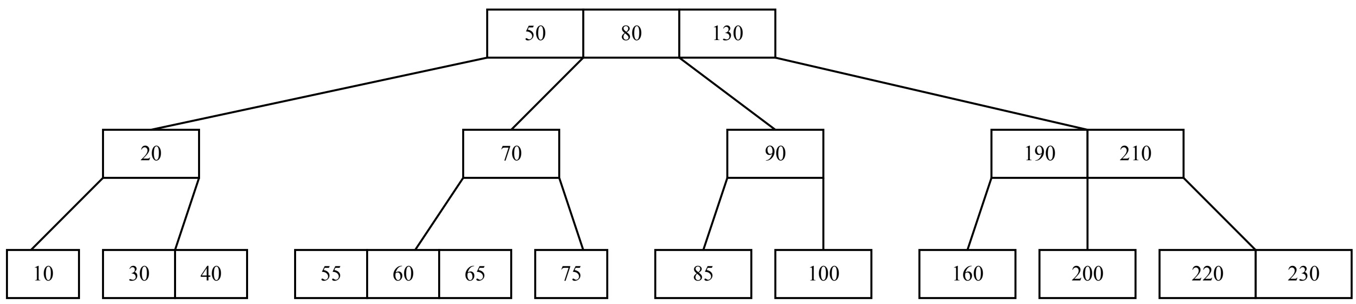
a) Obtener el árbol binario correspondiente a estos recorridos.
b) ¿Qué condiciones deben cumplir los elementos del recorrido Inorden para que el árbol resultante sea un árbol binario de búsqueda?
c) Manteniendo los mismos elementos, realiza los cambios oportunos en los recorridos para que el árbol resultante cumpla las condiciones de ABB y AVL.
d) Escribe el pseudocódigo del recorrido por niveles de un árbol binario.

3. Sea un montículo doble (DEAP) inicialmente vacio.

a) Inserta los siguientes ítems en el montículo doble vacio: 12, 19, 10, 16, 18, 17, 30, 22, 3, 21, 1.
b) Del montículo doble resultante del apartado a, extrae y borra el máximo y seguidamente el mínimo.
c) Explica cual es la complejidad temporal en el peor caso de insertar un elemento en un montículo doble. Razona tu respuesta. Explica cual es la complejidad temporal en el peor caso de la operación Borrar (ítem), que borra del montículo doble el ítem que pasamos por parámetro.

4. Dado el siguiente árbol 2-3-4:

a) Convertirlo a R-N. Para el caso de los 3-nodo, utilizar la conversión en la que el ítem izquierdo queda como hijo del ítem medio.
b) Sobre el árbol R-N resultado, insertar sucesivamente y en este orden las claves 67, 69 y 68, detallando los cambios de color y rotaciones empleadas. No será válido, realizar la inserción como si fuese un árbol 2-3-4 y realizar una transformación final a R-N, por ello, habrá que detallar claramente los cambios de color y rotaciones empleadas, en caso contrario no se puntuará la pregunta.



Examen PED julio 2011. Soluciones

1.

SINTAXIS
mezcla(cola, cola) → cola
Métodos auxiliares:
mezclaAux(cola, cola, cola) → cola
invierte(cola) → cola
invierteAux(cola, cola) → cola

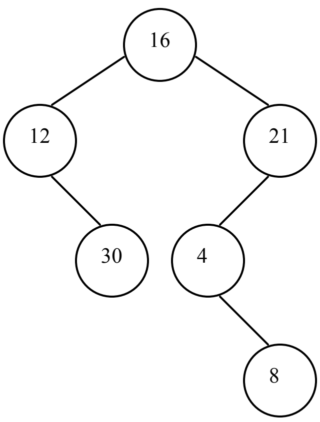
SEMÁNTICA
VAR c, d, e: cola; x, y: ítem;

mezcla(c, d) = mezclaAux(invierte(c), invierte(d), crear())
mezclaAux(crear(), crear(), c) = c
mezclaAux(crear(), encolar(c, x), d) = mezclaAux(crear(), c, encolar(d, x))
mezclaAux(encolar(c, x), crear(), d) = mezclaAux(c, crear(), encolar(d, x))
mezclaAux(encolar(c, x), encolar(d, y), e) = mezclaAux(c, d, encolar(encolar(e, x), y))

invierte(c) = invierteAux(c, crear())
invierteAux(crear(), c) = c
invierteAux(encolar(c, x), d) = invierteAux(c, encolar(d, x))

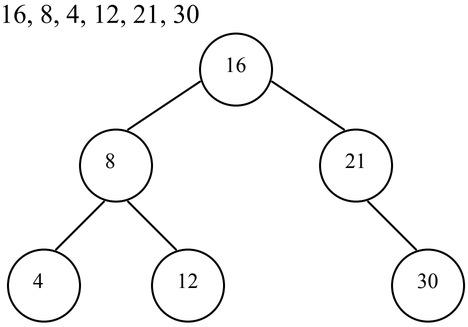
2.

a)



b) Los elementos deben estar ordenados de menor a mayor

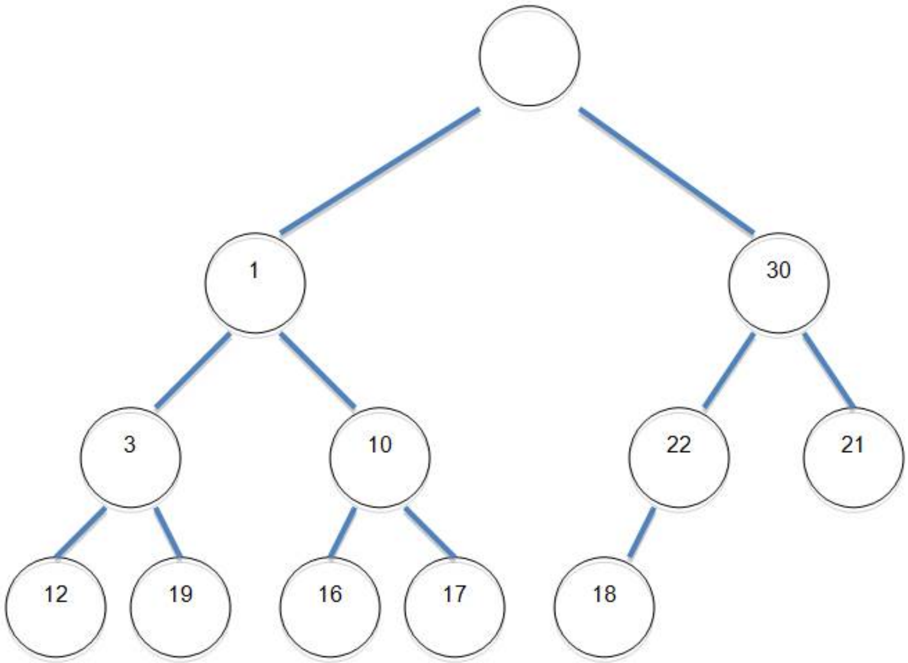
c) Inorden: 4, 8, 12, 16, 21, 30
Preorden: 16, 8, 4, 12, 21, 30



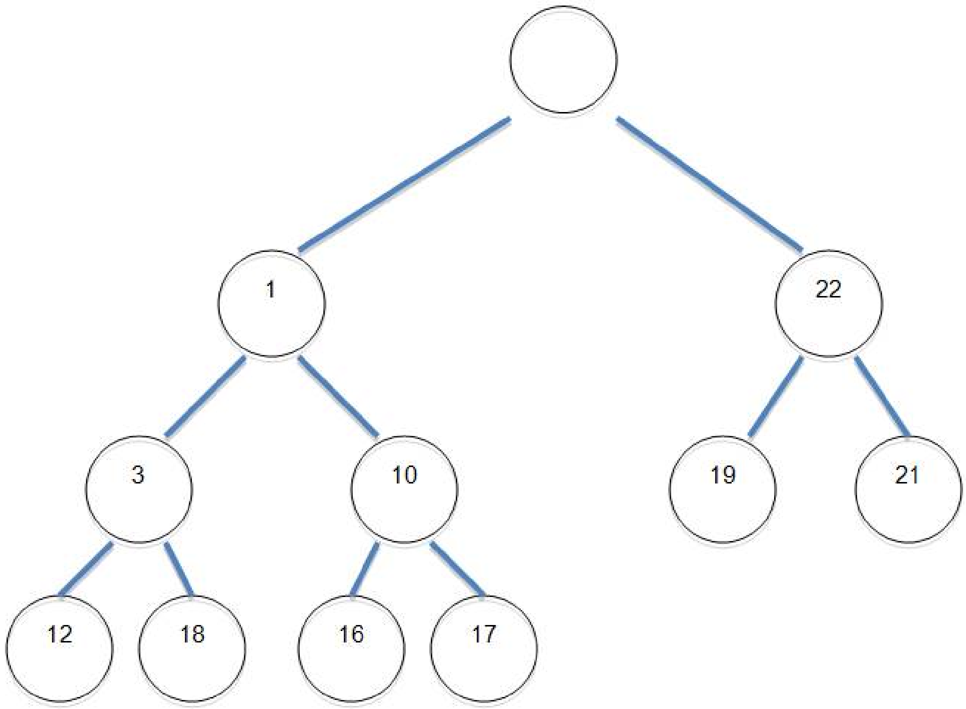
d)
```
algoritmo Niveles (a:arbin)
var c: cola de arbin; aux: arbin; fvar
Encolar(c, a)
mientras no EsVacia(c) hacer
    aux:=Cabeza(c)
    Escribe(Raiz(aux))
    Desencolar(c)
    si no EsVacio(HijoIz(aux)) entonces Encolar(c, HijoIz(aux)) fsi
    si no EsVacio(HijoDe(aux)) entonces Encolar(c, HijoDe(aux)) fsi
fmientras
falgoritmo
```

2.

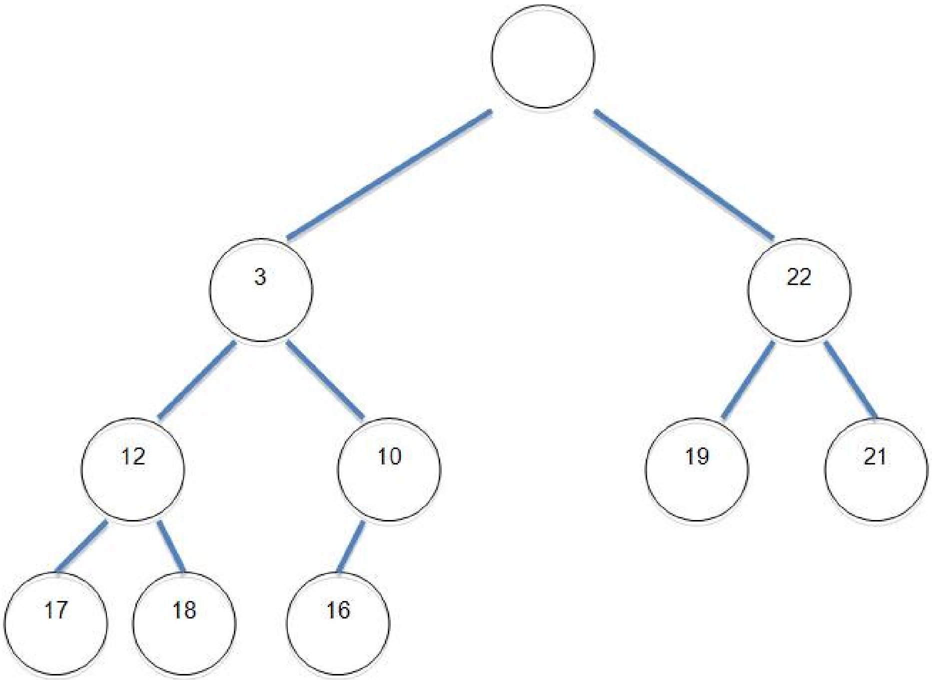
a)



b)
Borrar el 30



Borrar el 1

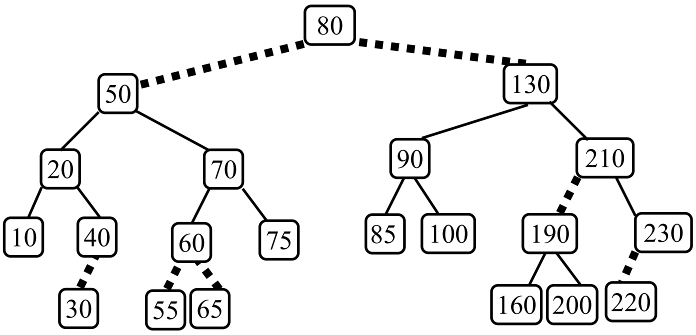


c) Al realizar una inserción, insertamos en el último elemento que sigue manteniendo la propiedad de árbol binario completo, por lo que estamos insertando en el último nivel del árbol. Para hacer las restructuraciones necesarias nos vamos a mover hacia arriba y en el peor de los casos llegar hasta la raíz. Como es un árbol completamente equilibrado, el numero de alturas que tenemos que recorrer en el peor caso son $h=\log_2 n$. Por lo tanto la complejidad es $O(\log_2 n)$

- La complejidad de la operación Borrar(ítem), en un montículo doble (DEAP) será en el peor caso O(n), ya que no sabemos la posición que ocupa el ítem y es posible que haya que borrar todos los ítems del montículo.
Si además quisiéramos mantener el resto de elementos en el montículo, es decir solamente borrar el elemento que pasamos por parámetro y no eliminar el resto, tendríamos que insertar en un montículo nuevo todos los elementos del montículo viejo, excepto el que queremos borrar. Esto tendría complejidad O(n + log n), en definitiva O(n)

4)

a)



b) Insertar 67: 2 cambios de color.
Insertar 69: Rotación II
Insertar 68: Cambio de color y Rotación ID