

Apellidos:

Nombre:

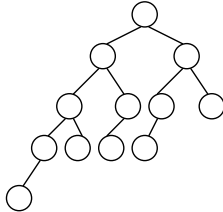
Convocatoria:

DNI:

Examen TAD/PED julio 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. A continuación comenzará el siguiente ejercicio.**
 - El test vale un 40% de la nota de teoría: 4 puntos.
 - En la **hoja de contestaciones** el verdadero se corresponderá con la **A**, y el falso con la **B**.

	V	F		
Dentro de la especificación algebraica de los números naturales definimos la sintaxis de la función F como: $F: \text{natural} \rightarrow \text{BOOL}$, y su semántica como: $F(\text{cero}) = \text{TRUE}$, $F(\text{suc}(\text{cero})) = \text{FALSE}$, $F(\text{suc}(\text{suc}(x))) = F(x)$. Para el número natural $x=35$, la función F devolvería FALSE.	<input type="checkbox"/>	<input type="checkbox"/>	1	V
En C++, la memoria que se reserva con new se libera automáticamente por el destructor.	<input type="checkbox"/>	<input type="checkbox"/>	2	F
La mejor complejidad temporal que se puede conseguir en un algoritmo es $O(n)$, con "n" como la talla del problema.	<input type="checkbox"/>	<input type="checkbox"/>	3	F
La semántica de la operación <i>sublista</i> del tipo lista vista en clase es la siguiente: $\text{VAR } L: \text{lista}; x, y: \text{item}; n: \text{natural}; p: \text{posicion};$ $\text{sublista}(L, p, 0) = \text{crear}()$ $\text{sublista}(\text{crear}(), p, n) = \text{crear}()$ $\text{si } p == \text{primera}(\text{inscabeza}(L, x)) \text{ entonces}$ $\quad \text{sublista}(\text{inscabeza}(L, x), p, n) = \text{inscabeza}(\text{sublista}(L, \text{primera}(L), n), x)$ $\text{si no sublista}(\text{inscabeza}(L, x), p, n) = \text{sublista}(L, p, n)$	<input type="checkbox"/>	<input type="checkbox"/>	4	F
Dado un único recorrido de un árbol binario lleno, es posible reconstruir dicho árbol	<input type="checkbox"/>	<input type="checkbox"/>	5	V
Cuando realizamos un recorrido en inorden en un árbol binario de búsqueda las etiquetas aparecen ordenadas de menor a mayor	<input type="checkbox"/>	<input type="checkbox"/>	6	V
Todo árbol binario de búsqueda es un árbol mínimo	<input type="checkbox"/>	<input type="checkbox"/>	7	F
El siguiente árbol está balanceado con respecto a la altura 	<input type="checkbox"/>	<input type="checkbox"/>	8	V
Todo árbol binario de búsqueda es un árbol 2-3.	<input type="checkbox"/>	<input type="checkbox"/>	9	F
En un árbol 2-3-4 sólo los nodos hoja y la raíz pueden ser de tipo 2-nodo	<input type="checkbox"/>	<input type="checkbox"/>	10	F
En un árbol rojo-negro la búsqueda de una etiqueta dependerá de los colores de los hijos de cada nodo	<input type="checkbox"/>	<input type="checkbox"/>	11	F
El árbol 2-3 es un árbol B m-camino de búsqueda con $m=2$	<input type="checkbox"/>	<input type="checkbox"/>	12	F
El TAD Diccionario es un subtipo del TAD Conjunto	<input type="checkbox"/>	<input type="checkbox"/>	13	V
La dispersión abierta elimina el problema del clustering secundario.	<input type="checkbox"/>	<input type="checkbox"/>	14	V
En un montículo doble de altura h se pueden almacenar un máximo de 2^{h-2} claves.	<input type="checkbox"/>	<input type="checkbox"/>	15	F
Un árbol leftist mínimo es un árbol binario mínimo tal que si no es vacío: $\text{CMÍN}(\text{HijoIzq}(x)) > \text{CMÍN}(\text{HijoDer}(x))$ para todo x no vacío	<input type="checkbox"/>	<input type="checkbox"/>	16	F

Examen PED julio 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.
 - **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. Realizar la especificación algebraica de la función $ABB(arbin) \rightarrow Bool$, la cual determina si el árbol binario que recibe como parámetro, es un árbol binario de búsqueda.

Notas:

- Un árbol binario es un árbol binario de búsqueda si al realizar el recorrido en inorden todos los elementos del árbol aparecen ordenados de menor a mayor.
- Se supone que está definida la siguiente operación: $inorden(arbin) \rightarrow Lista$.
- Se pueden utilizar funciones definidas en clase para una lista. Habrá que escribir la especificación algebraica del resto de funciones auxiliares que se utilicen.

2. Sea el siguiente recorrido PREORDEN de un árbol AVL: 10, 5, 3, 1, 4, 7, 16, 13, 11, 12, 14, 19, 21

- Reconstruye el árbol resultante a partir del recorrido Preorden anterior.
- En el árbol AVL resultante del apartado a), inserta los siguientes elementos: 20, 18, 17.
- Del árbol resultante del apartado B), borra los elementos 7, 21, 19.

Nota: Utiliza los criterios vistos en clase.

3.

- Construir un árbol de búsqueda digital con los elementos del 1 al 10 insertados en ese orden (1, 2, 3...10).
- Sobre el árbol de búsqueda digital anterior, calcular la complejidad temporal, en el peor de los casos, de las operaciones de inserción y búsqueda; dicha complejidad debe estar en función de la clave. Justifica la respuesta (en caso contrario, no se evaluará este apartado del ejercicio).

4. Dado el grafo dirigido representado por la lista de de adyacencia que se muestra a continuación:

- Obtened el árbol extendido en profundidad partiendo del vértice 1 y la clasificación de los arcos. Para la clasificación, los arcos se etiquetarán en la misma lista de adyacencia (**de otro modo no se corregirán**), por ejemplo:

A C
1 → 3 → 5

- Realizad el recorrido en anchura partiendo del vértice 1.
- Indica el menor número posible de arcos que habría eliminar para convertirlo en un grafo acíclico dirigido.
- Indica el menor número posible de arcos que habría eliminar para convertirlo en un árbol.

Nota: La lista de adyacencia de cada vértice se recorre en el mismo orden en que aparecen en la lista para todos los casos del ejercicio. Para vértices no visitados, continuar por el menor vértice.

1 → 3 → 5 → 8 → 10 → 11 → 12
2 → 1 → 4 → 8 → 10 → 11 → 12
3 → 2 → 4 → 10 → 11 → 12
4 → 5 → 1 → 8 → 10 → 11 → 12
5 → 2 → 3 → 10 → 11 → 12
6 → 9
7 → 6 → 9 → 5
8 → 5 → 3 → 10 → 11 → 12
9 → 6
10 →
11 →
12 → 11

Examen PED julio 2010. Soluciones

1.

ListaOrdenada(Lista) \rightarrow Bool

inorden(arbin) \rightarrow Lista

ABB(arbin) \rightarrow Bool

VAR x: item; L:Lista; i,d: Arbin;

ListaOrdenada(crear()) = TRUE

ListaOrdenada(incabeza(crear(), x) = TRUE

ListaOrdenada(incabeza(L,x)) =

Si (x < obtener(L, primera(L))

ListaOrdenada(L)

Sino

FALSE

Fsi

ABB(crear()) = TRUE

ABB(enraizar(crear(), x, crear()) = TRUE

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

si ListaOrdenada(inorden(enraizar(i, x, d)))

entones

TRUE

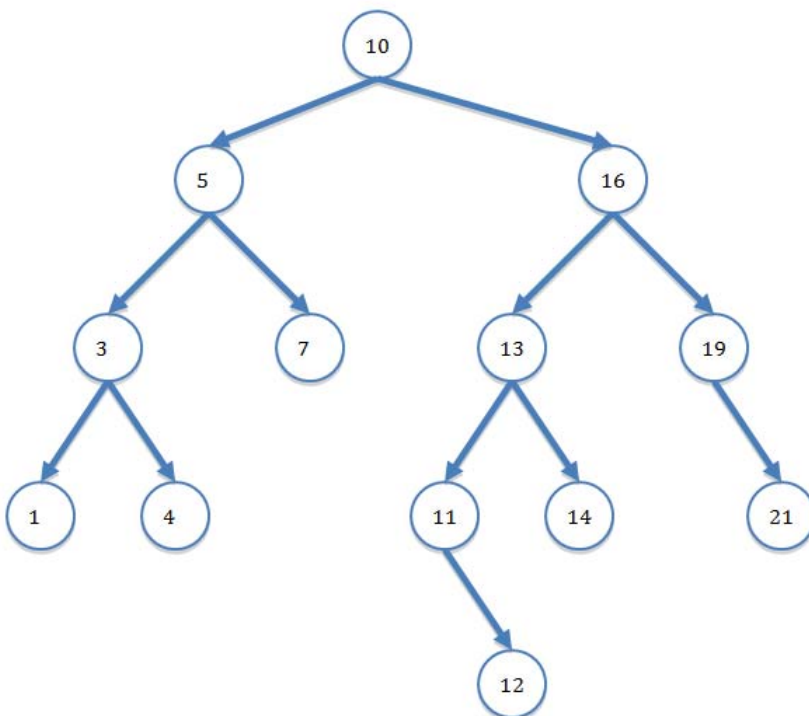
sino

FALSE

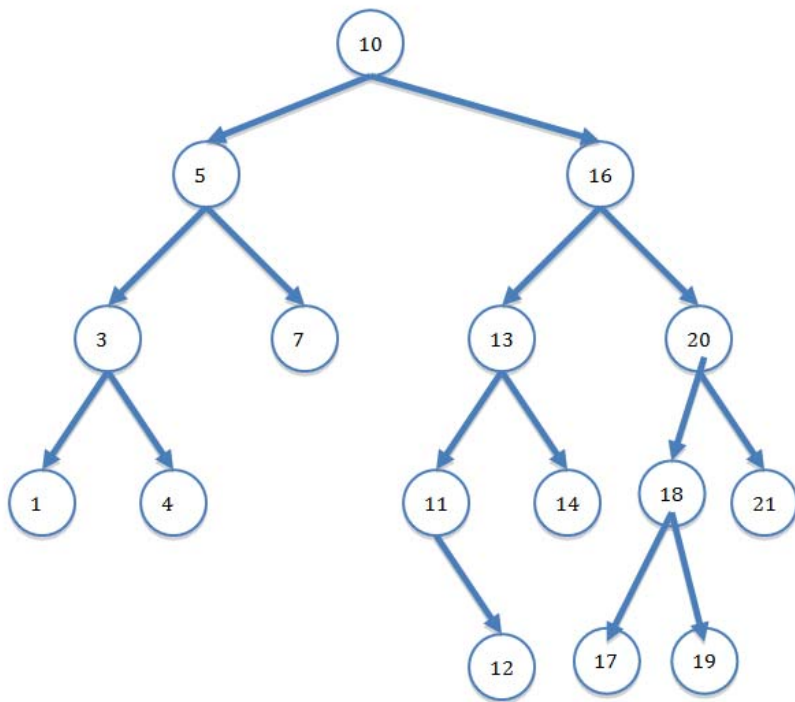
fsi

2.

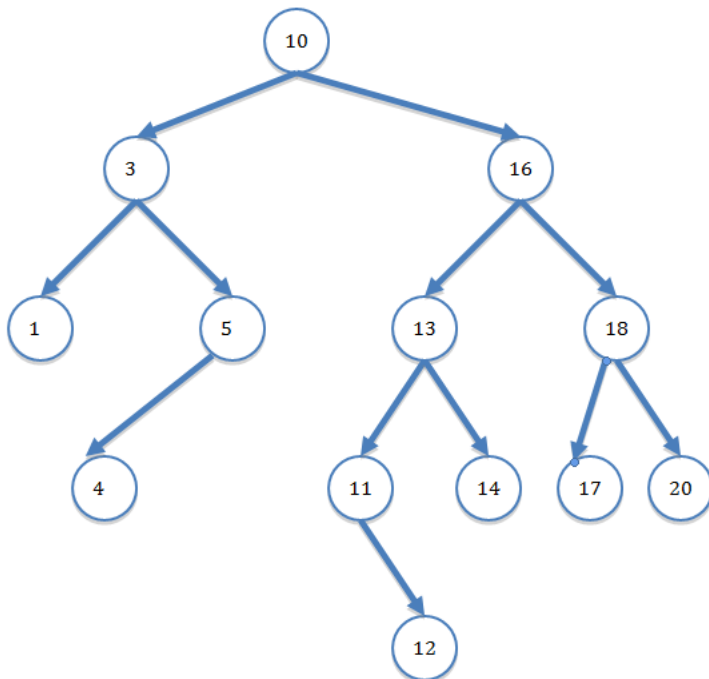
a)



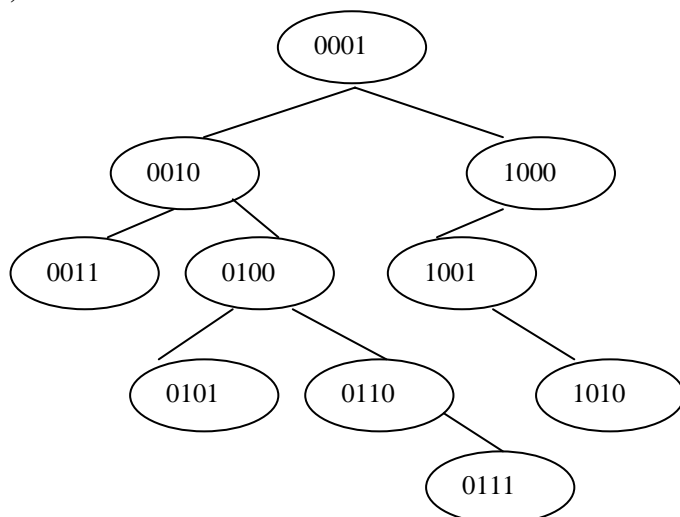
b)



c)



3. a)



- b)** Para la inserción y búsqueda $\rightarrow O(N+1)$ siendo N el número de bits de la clave, es decir $O(5)$.
 En el peor de los casos tenemos que recorrer la altura del árbol, que será el n° de bits de la clave más 1.
 Con 4 bits podemos representar $2^4 = 16$ elementos; para representar 16 elementos en un árbol binario, éste debe tener altura 5 (un árbol binario con altura h tiene como máximo $2^h - 1$ nodos; con altura 4 tendrá como máximo $2^4 - 1 = 15$ nodos; con altura 5 tendrá como máximo $2^5 - 1 = 31$ nodos).

4.

- a)** DFS = 1, 3, 2, 4, 5, 10, 11, 12, 8, 6, 9, 7

```

      A  AV AV  AV  AV  AV
1  → 3 → 5 → 8 → 10 → 11 → 12
      R  A  AV AV  AV  AV
2  → 1 → 4 → 8 → 10 → 11 → 12
      A  AV AV  AV  AV
3  → 2 → 4 → 10 → 11 → 12
      A  R  A  AV  AV  AV
4  → 5 → 1 → 8 → 10 → 11 → 12
      R  R  A  A  A
5  → 2 → 3 → 10 → 11 → 12
      A
6  → 9
      C  C  C
7  → 6 → 9 → 5
      C  R  C  C  C
8  → 5 → 3 → 10 → 11 → 12
      R
9  → 6

10 →
11 →
      C
12 → 11

```

- b)** BFS = 1, 3, 5, 8, 10, 11, 12, 2, 4
- c)** Los arcos de retroceso
- d)** Los arcos de retroceso, de avance y de cruce