Examen PED junio 2012.
Grado en Ingeniería Informática
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		
Longitud: LISTA -> NATURAL Si L es una lista, a es un item de la lista: a = Longitud (L) es un uso sintácticamente incorrecto de la operación	☑	☐	1	V
En layering los métodos de la clase derivada pueden acceder a la parte pública de la clase base.	☑	☐	2	V

En C++, el valor de la variable q al finalizar este fragmento de código es 7:
int q = 0;
int i;
for(i = 1; i < 5; i = i + 1)
  if(i != q)
    q += i;
	☑	☐	3	V

En la escala de complejidades se cumple que O(log n) ⊂ O(log log n).	☐	☑	4	F
En cualquier tipo de datos lineal cada elemento tiene un único sucesor y varios predecesor	☐	☑	5	F
Un árbol binario completo con n nodos y altura k es un árbol binario lleno para esa misma altura	☐	☑	6	F
El menor elemento en un árbol binario de búsqueda siempre se encuentra en un nodo hoja	☐	☑	7	F
Los árboles AVL son aquellos en los que el número de elementos en los subárboles izquierdo y derecho difieren como mucho en 1	☐	☑	8	F
Un árbol 2-3 es un árbol 2-ario de búsqueda	☐	☑	9	F
El árbol 2-3-4 no vacío tiene como mínimo una clave en cada nodo	☑	☐	10	V
En la representación de conjuntos mediante las listas el espacio es proporcional al tamaño del conjunto universal.	☐	☑	11	F
En el TAD Diccionario con dispersión abierta, la operación de búsqueda de una clave tiene una complejidad O(L), con L=longitud de la lista de claves sinónimas colisionadas.	☑	☐	12	V
El montículo o HEAP mínimo es un árbol binario lleno que además es árbol mínimo.	☐	☑	13	F
Un grafo no dirigido de n vértices es un árbol si está libre de ciclos y tiene “n-1” aristas	☑	☐	14	V
Al representar un grafo dirigido de N vértices y K aristas con una matriz de adyacencia, la operación de búsqueda de una arista tiene una complejidad de O(N).	☐	☑	15	F

Examen PED junio 2012. Grado en Ingeniería Informática

<u>**Normas:**</u>
- 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.
- 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.**
- **Las fechas de "Publicación de notas" y "Revisión del examen teórico" se publicarán en el Campus Virtual.**

1. LA PREGUNTA 1 SE CONTESTA EXCLUSIVAMENTE EN ESTA HOJA

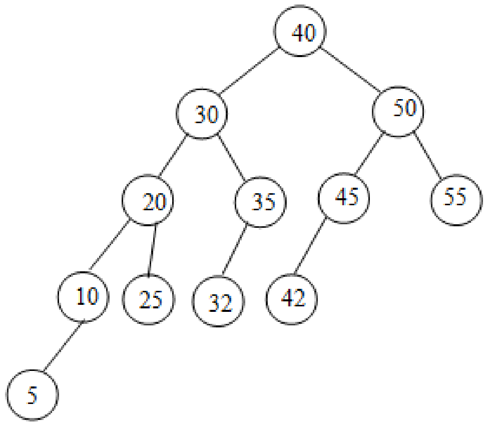
a) Utilizando exclusivamente operaciones constructoras generadoras, completa la semántica de la operación *borrar* de listas estudiada en clase.

Sintaxis: borrar(lista, posicion) → lista
VAR L_1, L_2: lista; x: item; p: posicion;
borrar(crear( ), p) = crear( )
si p == primera(inscabeza(L_1, x)) **entonces**
borrar( ) =
si no borrar( ) =

b) Utilizando exclusivamente operaciones constructoras generadoras, completa la semántica de la operación *multiplicación* de números naturales estudiada en clase.

Sintaxis: mult(natural, natural) → natural
VAR x, y: natural;
mult(cero, x) =
mult( ) =
mult( ) =

2. Dado el siguiente árbol AVL de entrada, efectuar los siguientes borrados en el árbol de forma consecutiva: 55, 32, 40, 30, 10 y 5. (Nota: al borrar un nodo con 2 hijos, sustituir por el mayor de la izquierda).



3.

a) Sea un árbol 234 inicialmente vacio. Insertar los elementos siguientes: 3, 30, 23, 6, 12, 18, 19, 31

b) Sea el árbol de la figura 1 un árbol 2-3-4. Borrar el ítem 20. Utilizad los siguientes criterios: Cr1. Si el ítem a borrar está en un nodo que no es una hoja sustituir por el ítem mayor de la izquierda. Cr.2. Si el nodo *q* tiene dos hermanos, el nodo *r* será el nodo de la izquierda.

c) Sea el árbol de la figura 1 un árbol 2-3. Borrar el ítem 20. Utilizad los siguientes criterios: Cr1. Si el ítem a borrar está en un nodo que no es una hoja sustituir por el ítem mayor de la izquierda. Cr.2. Si el nodo q tiene dos hermanos, el nodo r será el nodo de la izquierda.

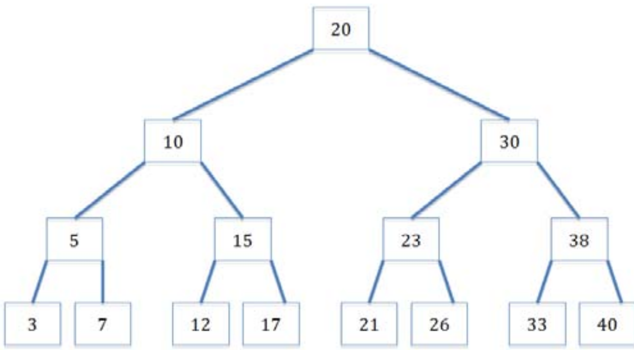


Figura 1

4. a) Aplicar el algoritmo *HeapSort* para ordenar el siguiente vector **de mayor a menor** (sin utilizar un vector auxiliar). En cada iteración del algoritmo, mostrar el montículo representado gráficamente y dentro del propio vector.

b) Indicar justificadamente el caso peor de la complejidad temporal de este algoritmo.

10	15	2	6	20	30	5	9	40	1

Examen PED julio 2012. Grado en Ingeniería Informática. Soluciones

1.

a)

borrar(crear( ), p) = crear( )

si p == primera(inscabeza(L_1, x)) **entonces**

borrar(inscabeza(L_1, x), p) = L_1

si no borrar(inscabeza(L_1, x), p) = inscabeza(borrar(L_1, p), x)

b)

mult(cero, x) = cero

mult(x, cero) = cero

mult(suc(y), x) = suma(mult(y, x), x)

2.

Borrado 55→ 2 rotaciones II

Borrado 32→ directo

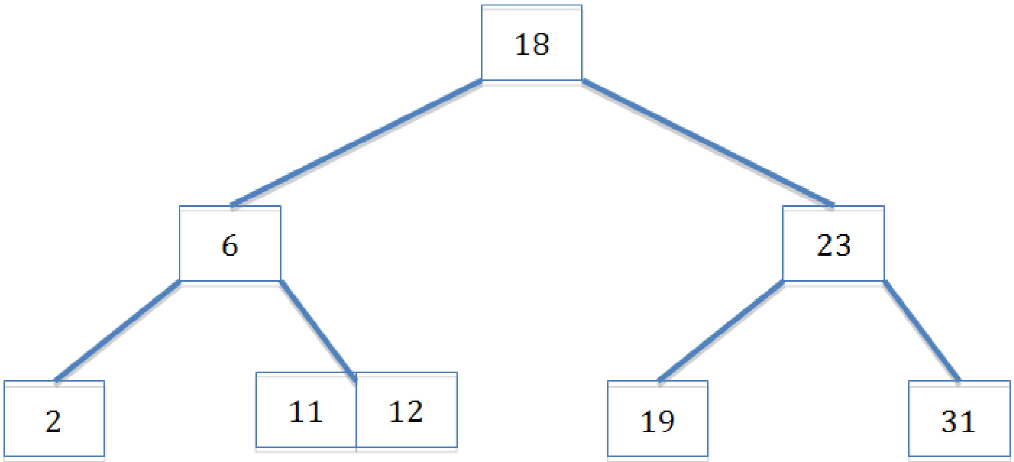
Borrado 40→ rotación DD

Borrado 30→ rotación II

Borrado 10 y 5→ rotación DI

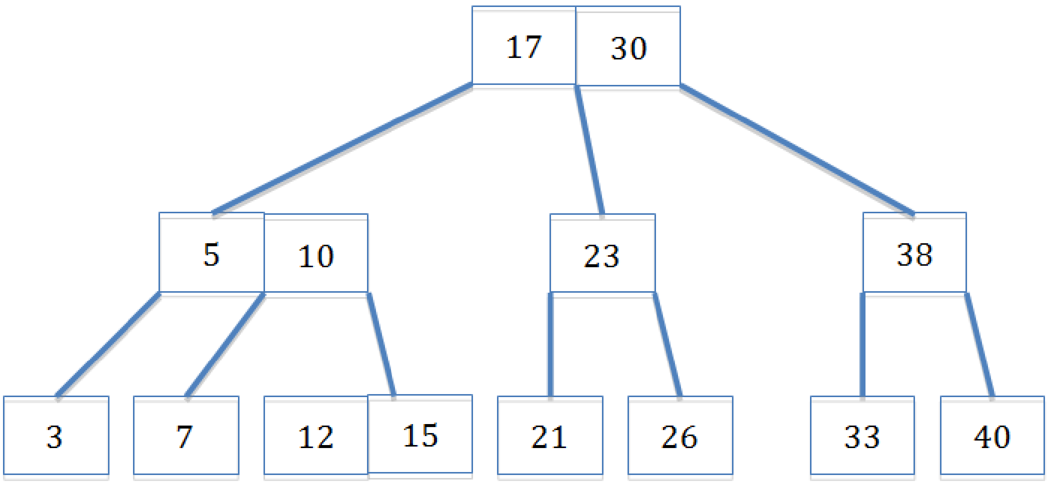
3.

a)



b)

c)



4. a) El algoritmo se aplicaría del siguiente modo:

1. Dejamos la parte izquierda del vector para obtener un montículo **mínimo** donde insertar todos los elementos del vector. La parte derecha se utilizará para almacenar los elementos todavía no insertados.
2. Cuando el montículo esté lleno, repetidamente se borra la raíz del montículo llevándola a la parte derecha del vector.
3. Al quedar vacío el montículo, el vector nos quedará ordenado de mayor a menor.

40	30	20	15	10	9	6	5	2	1

b) Con n el número de elementos del vector, se hacen $2n$ operaciones de insertar y borrar, cada una de ellas con coste logarítmico **(**dado que se realizan sobre un montículo, en el que estas operaciones tienen una complejidad lineal en función de la altura del árbol completo, por lo que es logarítmica en función del número de elementos del montículo), con lo que se concluye que tendrá un coste $O(n \log_2 n)$.