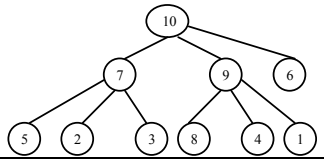
Apellidos:
Nombre:
Convocatoria:
DNI:

Examen PED diciembre 2009
Modalidad 0

Normas:
- La entrega del test no corre convocatoria.
- Tiempo para efectuar el test: **22 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		
Para el siguiente fragmento de código C++ de un posible método perteneciente a la conocida clase TCoordenada, la línea "delete b;" liberaría correctamente la memoria dinámica de *b*. *void Funcion(void) {* *TCoordenada *a = new TCoordenada;* *TCoordenada *b = new TCoordenada[5];* *(. . . . . .)* *delete b;* *}*	❑	❑	1	F
El resultado del cálculo de la complejidad temporal en el mejor caso de un algoritmo X, da como resultado n + n*log(n). Por lo tanto, diremos que la complejidad del algoritmo X cuando $n \to \infty$ pertenece a Ω (n).	❑	❑	2	F
Las pilas también se conocen como listas LIFO.	❑	❑	3	V
Dado un único recorrido de un árbol binario lleno, es posible reconstruir dicho árbol.	❑	❑	4	V
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, como mucho solo se va a efectuar una rotación.	❑	❑	6	V
La altura de un árbol 2-3 únicamente crece cuando se inserta un elemento y todos los nodos del árbol son 3-nodo.	❑	❑	7	F
Con las operaciones de inserción y borrado es posible conseguir un árbol 2-3-4 de altura 4 con todos sus nodos de tipo 2-nodo.	❑	❑	8	F
Las operaciones de transformación cuando se inserta un elemento en un árbol 2-3-4, en el caso de un árbol rojo-negro, se reducen a cambios de colores o rotaciones.	❑	❑	9	V
El árbol 2-3 es un árbol B m-camino de búsqueda con m=2.	❑	❑	10	F
La dispersión abierta elimina el problema del clustering secundario.	❑	❑	11	V
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 cuando se inserta el elemento.	❑	❑	12	F
El siguiente árbol es un montículo máximo:  	❑	❑	13	F
Para todo nodo de un árbol Leftist, se cumple que el número de nodos de su hijo izquierdo es mayor o igual que el de su hijo derecho.	❑	❑	14	F
Un grafo no dirigido de *n* vértices es un árbol si está libre de ciclos y tiene *n-1* aristas.	❑	❑	15	V

Examen PED diciembre 2009

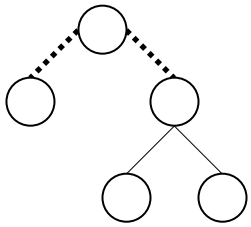
Normas:
- Tiempo para efectuar el ejercicio: **2 horas**
- En la cabecera de cada hoja **Y EN ESTE ORDEN** hay que poner: ***APELLIDOS, NOMBRE***.
- Cada pregunta se escribirá en hojas diferentes.
- Se dispone de 20 minutos para abandonar el examen sin que corra convocatoria.
- Las soluciones al examen 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**: el jueves 5 de noviembre. Revisión de exámenes el martes 10 de noviembre de 9:30 a 10:30 en la sala de reuniones 'Claude Shannon' (sótano de la EPS IV). Examen de prácticas el jueves 12 de noviembre de 15:00 a 17:00 en los laboratorios L25 y L27 de la EPS I.
- **Los alumnos que estén en 5ª o 6ª convocatoria deben indicarlo en la cabecera de todas las hojas**

1. A partir de la especificación algebraica de la cola, escribe la sintaxis y semántica de la operación *M()* que recibe dos colas y devuelve una cola nueva en la que se han encolado de forma alternada los elementos de las dos colas, empezando por la primera cola. Por ejemplo:

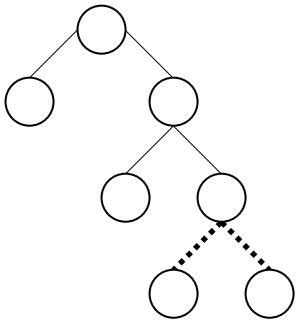
C1 = (a, b, c, d) C2 = (1, 2, 3)
M(C1, C2) = (a, 1, b, 2, c, 3, d)

2. De los siguientes árboles determinar cuáles son R-N razonando la respuesta:

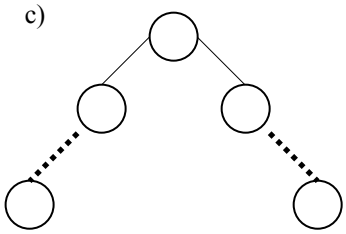
a)



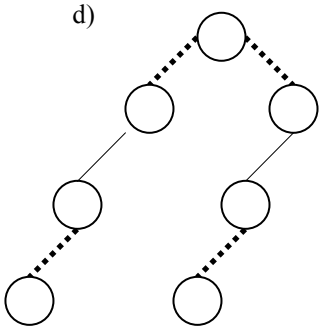
b)



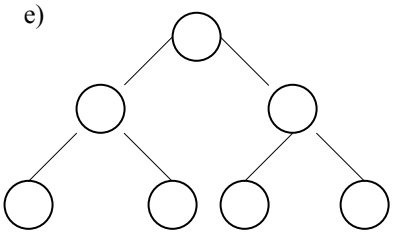
c)



d)



e)



3. Insertar en una tabla de dispersión cerrada de tamaño *B=7* los siguientes elementos: *10, 3, 17, 23, 21, 29 y 28*. Mostrar el cálculo para insertar cada elemento y la tabla final con la inserción de todos los elementos:

a) Con estrategia de redispersión aleatoria (c=2).
b) Con estrategia de redispersión *segunda función hash*.

4. Escribe en C++ la forma canónica de las clases de un trie (junto con las clases de los objetos que éste contenga) que tiene las siguientes complejidades para la operación de búsqueda de una palabra: Ω(1) y O(n * L), siendo *n* el número de letras diferentes que pueden formar parte de una palabra (por ejemplo las letras del alfabeto y los números), y *L* la longitud máxima de una palabra (por ejemplo de la palabra "maxilofacial", *L=12*).

NOTA:
- Hay que explicar en forma de comentarios cada campo privado de la clase.
- En la representación propuesta, explicar cuándo se producen los casos de complejidad mínima y máxima.
- El código C++ que se presente ha de ser directamente compilable, es decir, los errores de sintaxis de C++ se puntuarán negativamente.

Examen PED diciembre 2009. Soluciones

1.

** Sintaxis:
M(cola, cola) → cola
Aux1(cola, cola, cola) → cola
Aux2(cola, cola, cola) → cola

** Semántica:
Var c, c1, c2: cola; x: ítem;

M(c1, c2) = Aux1(crear(), c1, c2)

Aux1(c, crear(), crear()) = c
Aux1(c, encolar(c1, x), crear()) = Aux1(encolar(c, cabeza(c1)), desencolar(encolar(c1, x)), crear())
Aux1(c, crear(), encolar(c1, x)) = Aux1(encolar(c, cabeza(c1)), crear(), desencolar(encolar(c1, x)))
Aux1(c, encolar(c1, x), c2) = Aux2(encolar(c, cabeza(c1)), desencolar(encolar(c1, x)), c2)

Aux2(c, crear(), crear()) = c
Aux2(c, encolar(c1, x), crear()) = Aux2(encolar(c, cabeza(c1)), desencolar(encolar(c1, x)), crear())
Aux2(c, crear(), encolar(c1, x)) = Aux2(encolar(c, cabeza(c1)), crear(), desencolar(encolar(c1, x)))
Aux2(c, c1, encolar(c2, x)) = Aux1(encolar(c, cabeza(c2)), c1, desencolar(encolar(c2, x)))

Solución (otra forma):

** Sintaxis:
M(cola, cola) → cola
Concatena(cola, cola) → cola

** Semántica:
Var c1, c2: cola;

M(crear(), c1) = c1
M(c1, crear()) = c1
M(c1, c2) = Concatena(encolar(encolar(crear(), cabeza(c1)), cabeza(c2)), M(desencolar(c1), desencolar(c2))

Concatena(crear(), c1) = c1
Concatena(c1, crear()) = c1
Concatena(c1, c2) = concatena(encolar(c1, cabeza(c2)), desencolar(c2))

2.

a) No es R-N porque no todos los caminos desde la raíz a las hojas tienen el mismo número de nodos negros
b) Mismo razonamiento que a)
c) Sí es R-N, todos los caminos desde la raíz tienen el mismo número de nodos negros
d) Sí es R-N, mismo razonamiento que b)
e) Sí es R-N, todos los nodos del 2-3-4 equivalente son de tipo 2-nodo

3.

a) $h_i(x) = (H(x) + c \bullet i) \text{ MOD } B = (h_{i-1}(x) + c) \text{ MOD } B$
$H(10) = x \text{ MOD } B = 10 \text{ MOD } 7 = \mathbf{3}$
$H(3) = 3 \text{ MOD } 7 = 3$
$h_1(3) = (3+2) \text{ MOD } 7 = \mathbf{5}$
$H(17) = 17 \text{ MOD } 7 = 3$
$h_1(17) = (3+2) \text{ MOD } 7 = 5$
$h_2(17) = (5+2) \text{ MOD } 7 = \mathbf{0}$
$H(23) = 23 \text{ MOD } 7 = \mathbf{2}$
$H(21) = 21 \text{ MOD } 7 = 0$
$h_1(21) = (0+2) \text{ MOD } 7 = 2$
$h_2(21) = (2+2) \text{ MOD } 7 = \mathbf{4}$
$H(29) = 29 \text{ MOD } 7 = \mathbf{1}$
$H(28) = 28 \text{ MOD } 7 = 0$
$h_1(28) = (0+2) \text{ MOD } 7 = 2$
$h_2(28) = (2+2) \text{ MOD } 7 = 4$
$h_3(28) = (4+2) \text{ MOD } 7 = \mathbf{6}$
(0) 17 (1) 29 (2) 23 (3) 10 (4) 21 (5) 3 (6) 28

b) $k(x) = (x \text{ MOD } (B-1)) + 1$ // $h_i(x) = (H(x) + k(x) \bullet i) \text{ MOD } B = (h_{i-1}(x) + k(x)) \text{ MOD } B$
$H(10) = x \text{ MOD } B = 10 \text{ MOD } 7 = \mathbf{3}$

H(3) = 3 MOD 7 = 3
K(3) = (3 MOD 6) + 1=4
h_1(3)=(3+4) MOD 7=**0**
H(17) = 17 MOD 7 = 3
K(17) = (17 MOD 6) + 1=6
h_1(17)=(3+6) MOD 7=**2**
H(23) = 23 MOD 7 = 2
K(23) = (23 MOD 6) + 1=6
h_1(23)=(2+6) MOD 7=**1**
H(21) = 21 MOD 7 = 0
K(21) = (21 MOD 6) +1=4
h_1(21)=(0+4) MOD 7=**4**
H(29) = 29 MOD 7 = 1
K(29) = (29 MOD 6) +1=6
h_1(29)=(1+6) MOD 7=0
h_2(29)=(0+6) MOD 7=**6**
H(28) = 28 MOD 7 = 0
K(28) = (28 MOD 6) +1=5
h_1(28)=(0+5) MOD 7=**5**
(0) 3 (1) 23 (2) 17 (3) 10 (4) 21 (5) 28 (6) 29

4.

Caso Ω(1): al buscar la palabra "zoo" si en el trie solo hay almacenada esa palabra que empiece por "z"
Caso O(n*L): caso en que se busque una palabra de longitud *L* y en cada nodo haya que recorrer los *n* nodos de cada lista de strings.

```
class TNodoTrie;

class TListaNodo {
public:
        // Constructor por defecto
        TListaNodo();

        // Constructor de copia
        TListaNodo(const TListaNodo &);

        // Destructor
        ~TListaNodo();

        // Sobrecarga del operador asignación
        TListaNodo & operator=(const TListaNodo &);
prívate:
        // Letra o string a la que se corresponde el nodo
        char* c;

        // Será cierto si el camino desde la raíz hasta este nodo contiene una palabra almacenada en el trie
        bool pal;

        // Puntero al siguiente nivel del trie
        TNodoTrie* sigLetra;

        // Puntero al siguiente nodo de la lista
        TListaNodo* sig;
};

class TLista {
public:
        // Constructor por defecto
        TLista();

        // Constructor de copia
        TLista(const TLista &);
```

```
        // Destructor
        ~TLista();

        // Sobrecarga del operador asignación
        TLista & operator=(const TLista &);
prívate:
        // Puntero al primer nodo de la lista
        TListaNodo* primero;
};

class TNodoTrie {
public:
        // Constructor por defecto
        TNodoTrie();

        // Constructor de copia
        TNodoTrie(const TNodoTrie &);

        // Destructor
        ~TNodoTrie();

        // Sobrecarga del operador asignación
        TNodoTrie & operator=(const TNodoTrie &);
prívate:
        // Lista con cada uno de las posibles letras de una palabra
        TLista listaCaracteres;
};

class TTrie {
public:
        // Constructor por defecto
        TTrie();

        // Constructor de copia
        TTrie(const TTrie &);

        // Destructor
        ~TTrie();

        // Sobrecarga del operador asignación
        TTrie & operator=(const TTrie &);
prívate:
        // Puntero a la raíz del trie
        TNodoTrie* raiz;
};
```