/*

Ejercicio 1:

a) Dados dos nodos n1 y n2 en un árbol binario T y dadas las distancias (longitudes de los caminos) m1 y m2 de ambos nodos a su antecesor común más cercano (nodo más profundo que tiene tanto a n1 como a n2 como descendientes):

1-a: Si m1=m2=0 los nodos son el mismo nodo

1-b: Si m1=0 y m2>0: n2 es sucesor de n1

1-c: Si m1=m2=1 los nodos son hermanos;

1-d: Todo lo anterior es cierto <--- CORRECTA

b) Si inserto las claves {15, 7, 11, 23,18, 19, 9, 30, 1} en un AVL de enteros,

2-a: Hay que hacer dos rotaciones simples y una rotación doble

2-b: Hay que hacer tres rotaciones dobles,

2-c: Hay que hacer dos rotaciónes dobles

2-d: Todo lo anterior es falso <--- CORRECTA

c) Dados los siguientes recorridos en preorden = (C B F C I H G A J D E), y postorden = (F I C B H A D J E G C)

3-a: No hay ningún árbol binario con esos recorridos asociados; <--- CORRECTA

3-b: Hay 1 solo árbol binario con esos recorridos asociados:

3-c: Hay dos árboles binarios con esos recorridos asociados;

3-d: Hay múltiples árboles binarios con esos recorridos asociados;

d) Dado el siguiente fragmento de código: {map <int,int> M; M[0]=1; map <int,int> ::iterator p; p=M.find(7);}

¿Cual de las siguientes afirmaciones es verdadera?

4-a: M no se modifica y p->first=7

4-b: M se modifica y p ->first=7

4-c: Da un error

4-d: M se modifica y p=M.end() <--- CORRECTA

*/

[10:55]

/*

Ejercicio 1:

a) El TDA APO puede usarse para representar una cola con prioridad --> Verdadero

b) Puede hacerse la siguiente definiición priority_queue <int> :: iterator q; --> Falso

c) Un ABB puede reconstruirse de de forma unívoca dado su recorrido en inorden --> Falso

d) En un esquema de hashing cerrado, es correcto el uso como función hash de la función h(x)= M - (x% M) con M primo --> Falso

e) No puede construirse un AVL de profundidad 4 que contenga como etiquetas los números enteros del 1 al 12 --> Falso

*/

[11:00]

*

Ejercicio 1:

a) TDA cola con prioridad se puede implementar eficientemente con el uso de un heap(monton) [V]

b) Un AVL NO se puede reconstruir univocamente con el inorden

c) Es imposible que un arbol con mas de dos nodos sea un AVL y APO a la vez

b) La declaracion map<list<int>, string> m; es una declaración valida;

e) En un esquema de hashing double nunca puede ocurrir que para dos claves k1 y k2 distintas,

coincidan simultaneamente sus valores h (función hash primaria) y h0 (funcion hash secundaria). Es

decir, con k1!=k2, h(k1) = h(k2) y h0(k1) = h0(k2). <-- Falso

*/

[11:00]

/*

Ejercicio 1:

(a) La definición priority_queue<int>::iterator p; es correcta. <-- Falso

(b) Dado un árbol binario cuyas etiquetas están organizadas como un AVL, puedo

recuperarlo de forma unívoca a partir de su recorrido en inorden. <-- Falso

(c) El elemento de valor máximo en un ABB<int> se encuentra en el nodo más

profundo. <-- Falso

(d) Considerar un map <int, int> M en el que hacemos M[3]=7; Supongamos ==> (3,7) --> p.second = 9 --> (3,9)

ahora que hacemos

map <int, int> :: iterator p = M. find(3);

p → second =9;

Tras hacer eso, el valor de M[3] sigue siendo 7

Falso

(e) Si A es una tabla hash cerrada con un 50% de elementos vacíos y un 40% de elementos borrados y B una tabla hash cerrada con un 50% de elementos vacíos y sin elementos borrados, A y B son igual de eficientes cara a la búsqueda de un elemento. <-- Falso

*/