Universidad de Granada

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN E INTELIGENCIA ARTIFICIA

ETSII Informática y de Telecomunicación, C/ Periodista Daniel Saucedo Aranda s/n- 18071- Granada (España)

Estructuras de Datos
Curso 2017-2018. Convocatoria de Enero
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas

1. (0.5 puntos) (a) Razonar la verdad o falsedad de las siguientes afirmaciones:

   (a) El orden en que las hojas se listan en los recorridos preorden, inorden y postorden de un árbol binario es el mismo en los tres casos.

   (b) Un ABB puede reconstruirse de forma unívoca dado su recorrido en preorden

   (c) Un APO puede reconstruirse de forma unívoca dado su recorrido en postorden

   (d) Es correcto en un esquema de hashing cerrado el uso como función hash de la función h(k)= [k + random(M)] % M, M primo y con random(M) una función que devuelve un número entero aleatorio entre 0 y M-1

   (e) Es correcto en un esquema de hashing cerrado el uso como función hash secundaria de la función h″Ḥ(x)=[(B-1)-(x % B)] % B con B primo

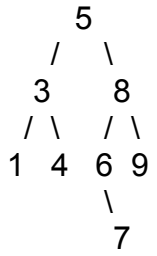
2. (1.5 puntos) Supongamos que tenemos una clase **Liga** que almacena los resultados de enfrentamientos en una liga de baloncesto:

```
struct enfret{
   unsigned char eq1,eq2; //codigos de los equipos enfrentados
   unsigned int canastas_eq1,canastas_eq2; //canastas por cada equipo
};
class Liga{
 private:
   list<enfret> res;
…
};
```

- Implementar un método que dado un código de equipo obtenga el número de enfrentamientos que ha ganado.
- Implementar la clase iterator dentro de la clase Liga que permita recorrer los enfrentamientos en los que el resultado ha sido el empate. Implementar los métodos begin() y end().

3. (1 punto) Implementar una función **int orden (list<int> L);** que devuelva 1 si L está ordenada de forma ascendente de principio a fin, 2 si lo está de forma descendente y 0 si no está ordenada de ninguna forma.

4. (1 punto) Dado un árbol binario de búsqueda, implementar una función para imprimir las etiquetas de los nodos en orden de mayor profundidad a menor profundidad. Ejemplo:



El resultado seria 7,1,4,6,9,3,8,5.

5. (1 punto) Tenemos un contenedor de pares de elementos, {clave, ArbolBinario<int>} definida como:

```
template <typename T>
class contenedor{
    private:
      unordered_map<T, ArbolBinario<int> > datos;
    ……..
    ……..
}
```

Implementar un iterador que itere sobre los elementos que cumplan la propiedad de que la suma de los elementos del ArbolBinario<int> sea un número par. Han de implementarse (aparte de las de la clase iteradora) las funciones begin() y end().

6. (1 punto) Un "heap-doble" es una estructura jerárquica que tiene como propiedad fundamental el que para cualquier nodo Z a profundidad **par** la clave almacenada en Z es **menor** que la del padre pero **mayor** que la del abuelo (cuando existen), y para cualquier nodo Z a profundidad **impar**, la clave almacenada en Z es **mayor** que la del padre pero **menor** que la del abuelo (cuando existen), siendo el árbol binario y estando las hojas empujadas a la izquierda. Diseñar una función para insertar un nuevo nodo en la estructura y aplicarla a la construcción de un heap-doble con las claves {30, 25, 12, 16, 10, 15, 5, 18, 23, 32, 4, 17}.