UNIVERSIDAD DE GRANADA
Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2018-2019
Convocatoria de Enero
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

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

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

   (b) Dado un árbol binario cuyas etiquetas están organizadas como un **AVL**, puedo recuperarlo de forma unívoca a partir de su recorrido en inorden.

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

   (d) Considerar un **map <int, int> M** en el que hacemos **M[3]=7;** Supongamos 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

   (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.

2. (1.5 puntos) Implementa una **clase documento** a partir de una lista de palabras que componen un texto. Esta clase guardaría de forma eficiente las posiciones en las que se encuentran cada una de las palabras que forman dicho texto.

   a) Indica una representación e implementa el constructor

      documento::documento(const list<string> &texto)

   b) Implementa una función que devuelva de forma eficiente las posiciones en las que aparece una determinada palabra.

      set<int> documento::posiciones(string palabra) const;

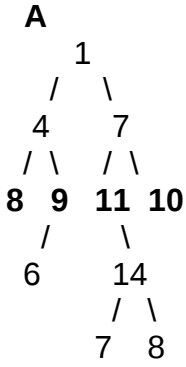
   c) Implementa una función que calcule la palabra que se encuentra en la posición iésima

      string documento::palabra(int i) const;

3. (1 punto) Dada una lista de enteros L y 2 listas seq y reemp, implementa una función:
   void reemplaza (list<int> & l, const list<int> &seq, const list<int> &reemp);
   que busque todas las secuencias seq en l y las reemplace por reemp.
   P.ej si L={1,2,3,**4,5,1**,2,3,**4,5,1**,2,3,4,5} , seq= {4,5,1} y reemp ={9,7,3, 5}
   L quedaría como L={1,2,3,**9,7,3, 5**,2,3,**9,7,3, 5**,2,3,4,5}

4. (1 punto) Implementa una función
   list<int> nivel (const bintree<int> & A);
   que dado un árbol binario A, devuelva una lista con las etiquetas del nivel que tenga un mayor número de nodos

Ejemplo:



L={8,9,11,10}

En caso de que haya más de una lista solución, basta con devolver una de ellas

5. (1 punto) Detalla cada una de las operaciones siguientes:
   a) Insertar las claves {4, 12, 16, 37, 6, 58, 23, 61, 9, 10} en una **Tabla Hash cerrada** de tamaño 13. A continuación borrar el 10 y el 37 y finalmente insertar el valor 48. Resolver las colisiones usando **rehashing doble.**
   b) Construir un **AVL** insertando, en ese orden, las siguientes claves {99, 28, 70, 81, 47, 38, 100, 21, 16, 49, 57}, especificando los pasos seguidos e indicando cuando sea necesario el tipo de rotación que se usa para equilibrar.
   c) Construir un **árbol parcialmente ordenado** realizando las siguientes tareas:
      - Inserta, en ese orden, las siguientes claves {10, 5, 12, 12, 5, 3, 9, 4, 3}
      - Borrar tres elementos.

6. (1 punto) Tenemos un contenedor de pares de elementos, {clave, list<int>} definida como:

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

Implementa un **iterador** que itere sobre los elementos que cumplan la propiedad de que la lista asociada a una clave solo contenga números pares. Se debe implementar (aparte de las de la clase iteradora) las funciones begin() y end().

Tiempo: 3 horas

UNIVERSIDAD DE GRANADA
Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2018-2019
Convocatoria de Enero
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Preguntas específicas para aquellos estudiantes sujetos a convocatoria adicional o evaluación única final (4 puntos)

1.- (2 puntos) Sea el **TDA editor** que mantiene un conjunto de lineas de texto. Cada línea viene delimitada por un retorno de carro. Las operaciones que se pueden realizar son

- insertar una nueva linea de texto,
- borrar la última línea de texto añadida,
- deshacer todo lo que se ha hecho desde un borrado previo (undo),
- devolver todas las líneas insertadas desde las más antigua hasta la más moderna.

Realiza las siguientes tareas:

a) Indica una representación eficiente para el TDA editor,
b) Implementa las funciones:
   - **void editor::insertar(string linea)**
   - **string editor:borrar_ultima()**
   - **void editor::deshacer()**
   - **list<string> editor::contenido().**

NOTA: Se pueden deshacer consecutivamente varios borrados que se hayan podido realizar previamente.

2.- (1 punto) Implementa una función

int max_subtree (bintree<int> & T)

que devuelva la suma de etiquetas máxima de entre todos los posibles subárboles del árbol binario T (las etiquetas de los nodos pueden ser positivas o negativas y por tanto el subárbol vacío puede ser el de mayor suma que se considera (en ese caso 0)

3.- (1 punto) En un hospital quieren implementar un sistema que permita que se puedan atender a usuarios considerando la gravedad. Además se debe poder hacer una búsqueda sobre los datos de cada paciente en función de su DNI o de su nombre completo. Indica una representación adecuada e implementa las operaciones:

- **urgencias::insertar_paciente(string dni, string nombre, string apellidos, int gravedad)**
- **urgencias::cambiar_gravedad(string dni, int nueva_gravedad)**

Tiempo: 1 hora