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

Estructuras de datos. Curso 2019-2020
Convocatoria ordinaria de Enero
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

1. (0.75 puntos) Elegir la opción correcta en cada uno de los siguientes supuestos (y explica brevemente el porqué):

   (1) 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

   (2) 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

   Mostrar el arbol final

   (3) 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; **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

   (4) 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()

2. (1.5 puntos) Se dispone de un vector de gran tamaño donde se almacenan claves de tipo char ordenadas que pueden aparecer de forma consecutiva un numero indefinido de veces. P.ej:

   ```
   { a,a,a,a,a,b,b,b,c,c,c,c,c,c,c,d,d,d,d,d,d,.....}
   ```

   (a) Implementar un TDA (basado en el tipo map de la STL) capaz de almacenar los datos del vector e implementar una función que dada una posición i del vector original devuelva el valor almacenado en esa posición.

   (b) Implementar un **iterador** que itere sobre los elementos que cumplan la propiedad de que su número de repeticiones sea un número par. Se deben implementar (aparte de las de la clase iterator) las funciones begin() y end().

3. (1 punto) Implementar una función

   int minsum(const list<int> &L);

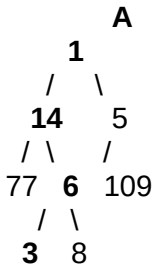
   que dada una lista de enteros L, devuelva la suma total de los enteros de la sublista de menor suma.
   Ejemplos:
   L=[1 -2 3 -3 -1 2 5]; devuelve: -4 (corresponde a la sublista [-3 -1])
   L=[1 -2 1 -3 -1 2 5]; devuelve: -5 (corresponde a la sublista [-2 1 -3 -1])

4. (1 punto) Implementar una función

   list<int> caminodemenores (const bintree<int> & A);

   que dado un árbol binario A, devuelva en L el camino de la raíz a una hoja de forma que la suma de sus etiquetas sea la menor posible. No se pueden usar iteradores.

   Ejemplo:



   L={1,14,6,3}

5. (1 punto) Implementar una función

   int mas_conectado(const vector< set<int> > &VS);

   que devuelve el índice **j** tal que el conjunto **VS[j]** es el conjunto que está conectado con un mayor número de otros conjuntos de VS. Decimos que dos conjuntos están conectados si no son disjuntos, es decir, si tienen intersección común no vacía. En el caso de haber varios conjuntos con el mismo número de conexiones debe devolver el primero de ellos.
   Ejemplos:
   VS=[{0},{1},{2},{0,1,2}], devuelve 3
   VS=[{0,1,2},{0},{1},{2}], devuelve 0
   VS=[{0,6,9},{5,6,9},{5},{1},{5,9},{5},{1,5,7}], devuelve1

6. (0.75 puntos) Insertar (detallando los pasos) las siguientes claves (en el orden indicado):

   {47, 31, 49, 66 ,50 ,52, 82, 38, 7, 63, 53}

   en una **tabla hash cerrada** de tamaño 13 con resolución de colisiones usando hashing doble. Insertar los elementos anteriores en un **APO**. Borrar 1 elemento del APO resultante.

Tiempo: 3 horas

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2019-2020
Convocatoria ordinaria 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) Supongamos que dos personas (usuarios) se insertan en un laberinto de habitaciones por la misma puerta. El laberinto tiene las siguientes características:

- Cada habitación del laberinto tiene una puerta por donde se entra y dos puertas posibles por la que se sale.
- Habrá habitaciones que conduzcan a la calle y por lo tanto finaliza el recorrido por el laberinto.
- Para pasar por una puerta se tira un dado y si sale un número par se sale por la puerta izquierda y si sale impar se sale por la derecha.
- A cada habitación solamente se accede por una única habitación.

Se pide:
- Establece la representación más eficiente para el laberinto y usuario.
- Implementa el constructor de la clase. Debe construir un laberinto con n habitaciones y con las características indicadas.
- Implementa la función que dado un laberinto y dos usuarios establezca de entre estos dos usuarios cual de ellos llega antes a la calle.

2.- (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

a) insertar una nueva linea de texto,
b) borrar las n últimas líneas de texto añadidas,
c) deshacer todo lo que se ha hecho desde un borrado previo,
d) devolver todas las líneas insertadas desde las más antigua hasta la más moderna.

Realiza las siguientes tareas:

e) Indica una representación eficiente para el TDA editor,
f) Implementa las funciones:
   - **void editor::insertar(string linea)**
   - **list<string> editor:borrar(int n_lineas)**
   - **void editor::deshacer()**
   - **list<string> editor::contenido().**

Tiempo: 1 hora