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

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

1. (1 punto) Implementar una función:

   void juntalista (list<int> &L, int n)

   que dada una lista L, agrupe los elementos de n en n dejando su suma

   P.ej si L={**1,3,2,**4,5,2,**2,3,5**,7,4,3,**2,2**} y **n=3** quedaría L={**6,**11,**10**,14,**4**}

   No pueden usarse estructuras auxiliares. Si n=0 devuelve la misma lista y si L está vacia, devuelve la lista vacia

2. (1 punto) En un hospital quieren implementar un sistema que permita que se pueda atender a usuarios considerando su 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:

   void urgencias::insertar_paciente(string dni, string nombre, string apellidos, int gravedad)

   void urgencias::cambiar_gravedad(string dni, int nueva_gravedad)

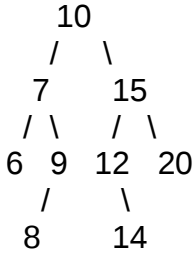
3. (1 punto) Implementar una función

   bool esABB (bintree <int> & A,);

   que devuelva true si el arbol binario A es un ABB y false en caso contrario

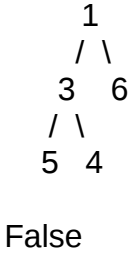
Ejemplos:

A



True

A



False

4. (1 punto) Implementar una función

   bool inall (list<set<int> > & LS, set<int> & S);

   que devuelva true si algún conjunto está incluido en todos los demás y tal conjunto lo devuelva en S

   P.ej: Si LS = [{1,2,3}, {2,3,4}] devuelve FALSE

   Si LS= [{**1,2,3**}, {**1,2,3**,4}, {**1,2,3**}] devuelve TRUE y S={**1,2,3**}

   Si LS= [{**1**,2,3}, {**1**}, {**1**,2}] devuelve TRUE y S={**1**}

5. (1 punto) Detalla cada una de las operaciones siguientes:
   a) Insertar las claves {5, 13, 17, 38, 7, 59, 24, 62, 10, 11} en una **Tabla Hash cerrada** de tamaño 13. A continuación borrar el 10 y el 38 y finalmente insertar el valor 48. Resolver las colisiones usando **rehashing doble**.
   b) Construir un **AVL** insertando, en ese orden, las siguientes claves {98, 27, 69, 80, 46, 37, 99, 20, 15, 48, 56}, especificando los pasos seguidos e indicando cuando sea necesario el tipo de rotación que se usa para equilibrar.
   c) Construir un **APO-min** realizando las siguientes tareas:
      - Insertar, en ese orden, las siguientes claves {11, 6, 13, 14, 7, 4, 10, 5, 8}
      - Borrar dos elementos.

6. (1 punto) Tenemos un contenedor de pares de elementos, {string set<int>} definido como:

```
class contenedor {
    private:
        map<string, set<int> > datos;
    ……..
    ……..
}
```

Implementar un **iterador** que itere sobre los elementos que cumplan la propiedad de que el conjunto asociado solo contenga elementos mayores que 20. 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 Febrero
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Preguntas específicas de prácticas (4 puntos)

1.- (3 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 la última línea de texto añadida,
c) deshacer todo lo que se ha hecho desde un borrado previo (undo),
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)**
   - **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)

Tiempo: 1 hora