UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

1. (1 punto) **(a)** Si inserto las claves {18, 10, 14, 26, 12, 21, 33, 4} en un AVL de enteros, (a1)**:** Hay que hacer dos rotaciones simples y una rotación doble (a2)**:** Hay que hacer una rotación simple y una rotación doble, (a3)**:** Hay que hacer dos rotaciones dobles y una rotación simple (a4)**:** Todo lo anterior es falso. **Mostrar el árbol final**

   (b) ¿Puede reconstruirse forma unívoca un árbol binario en el que todos los nodos tienen 0 ó 2 hijos, conociendo su preorden? ¿y un árbol binario completo conociendo su postorden? Razonarlo

   (c) Dados los siguientes recorridos Preorden y Postorden:

   Pre = {A,Z,W,R,V,X,Q,T,Y,L} **Post = {R,V,W,Q,X,Z,Y,L,T,A}**

   (c1) Hay exactamente 2 árboles binarios con esos recorridos. (c2) No hay ningún árbol binario con esos recorridos.(c3) Hay exactamente 1 árbol binario con esos recorridos. (c4) Hay más de 2 árboles binarios con esos recorridos

   (d) Dado el siguiente fragmento de código:

   {**map <int,int> M**; M[0]=1; map <int,int> ::iterator p; p=M.find(9);}

   ¿Cual de las siguientes afirmaciones es verdadera?

   d-1: M no se modifica y p->first=9 **d-3:** M se modifica y p ->first=9

   d-2: Da un error **d-4:** M se modifica y p=M.end()

2. (1 punto) Tenemos un contenedor de pares de elementos, **{clave, bintree<int>}** definida como:

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

   Implementar un **iterador** que itere sobre las claves que cumplan la propiedad de que el bintree asociado tenga como suma de sus etiquetas un número par. Se deben implementar (aparte de las de la clase iterator) las funciones begin() y end() de la clase contenedor.

3. (1 punto) Implementar una función

   bool permutalista (list<int> & L1, list<int> & L2)

   que devuelva true si L1 y L2 tienen la misma cantidad de elementos y los elementos de L1 son una permutación de los de L2

   P.ej: si L1={1,23,21,4,2,3,0} y L2={21,1,3,2,4,23,0} devolvería TRUE pero si L1={1,3,5} y L2={1,5,4} devolvería FALSE

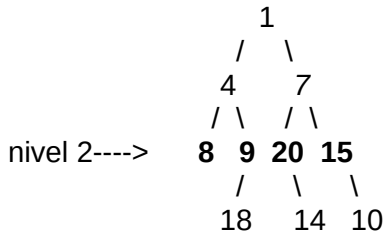
   Si hay elementos repetidos tienen que estar el mismo número de veces en las 2 listas para poder ser TRUE. **No pueden usarse estructuras auxiliares**, el algoritmo puede ser destructivo y no conservar las listas iniciales y **no puede usarse ningún algoritmo de ordenación**.

4. (1 punto) Dado un bintree<int>, implementar una función

void prom_nivel(bintree<int> &T, list<float> &P);

que genere una lista de reales P, donde el primer elemento de la lista sea el promedio de los valores de las etiquetas de los nodos del árbol de nivel 0, el segundo sea el promedio de los de nivel 1, el tercero el promedio de los de nivel 2, y así sucesivamente. Es decir, que si el árbol tiene profundidad N, la lista tendrá N+1 elementos de tipo float.

Ejemplo: **A**



prom_nivel(A, P)

P=**{**1.0, 5.5, **13.0**, 14.0**}**

5. (1 punto) Implementar una función:

bool tiene_suma_constante (set<int> & s, int M);

que dado un conjunto de enteros **s** y un entero **M**, devuelva true si existe un subconjunto de elementos enteros de s cuyos valores sumen exactamente M.

P.ej si. s={**1,2,3,4,5**,6}; **tiene_suma_constante**(s,**15**) devolvería **true**
y si hacemos **tiene_suma_constante**(s,**22**) devolvería **false**

6. (1 punto) (a) Insertar en el orden indicado (detallando los pasos) las siguientes claves en un **APO**: {10, 5, 12, 12, 5, 3, 9, 4, 3}. Borrar dos elementos y mostrar el APO resultante

(b) 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.

Tiempo: 2.30 horas

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

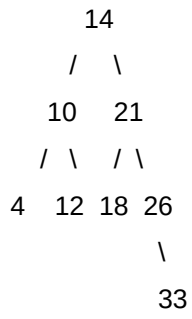
Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

SOLUCIONES EXAMEN CONVOCATORIA FEBRERO 2023

Pregunta 1

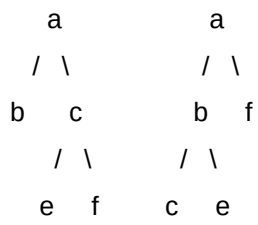
1.a Respuesta **(a4). Todo lo anterior es falso**. Para construir un AVL con esas claves se necesitan exactamente 2 rotaciones dobles (una a la derecha sobre el nodo 18 y otra a la izquierda también sobre el nodo 18) y nada más.

El árbol resultante es:



1.b. No puede reconstruirse de forma únivoca un árbol homogeneo a partir del preorden.

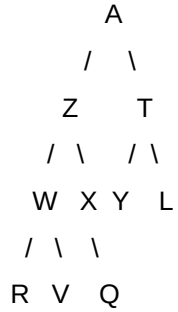
Contraejemplo:



Ambos son homogeneos y tienen el mismo preorden: {a,b,c,e,f}

En cambio **un árbol completo si puede reconstruirse a partir de su postorden** porque tiene configurada a priori su estructura geométrica a partir de conocer en número de nodos que tiene ese postorden

1.c La respuesta correcta es la (c1). **Hay exactamente 2 árboles binarios con esos recorridos.**



En función de que Q se sitúe a la derecha o a la izquierda de X, resultan justo 2 árboles diferentes

1.d. La respuesta correcta es la d-4: **M se modifica y p= M.end();**

Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Pregunta 2

```
#include <unordered_map>
#include "bintree.h"
using namespace std;

int Suma (bintree<int>::node n){
  if (n.null())
    return 0;
  else
    return (*n)+Suma(n.left())+Suma(n.right());
}

template <typename T>
class contenedor{
 private:
   unordered_map<T,bintree<int> > datos;

 public:
  //...
  class iterator{
  private:
    typename unordered_map<T,bintree<int> >::iterator it,final;
  public:
    iterator(){}

     bool operator==(const iterator &i)const{
            return i.it==it;
     }

     bool operator!=(const iterator &i)const{
            return i.it!=it;
     }

     pair<const T, bintree<int> > & operator* (){
           return (*it);
     }
```

```
 iterator & operator++(){
            ++it;
            bool salir =false;
            while (it!=final && !salir){
              if (suma((*it).second.root())%2==0)
                    salir =true;
              else ++it;
            }
            return *this;
    }
     friend class contenedor;
};

 iterator begin(){
   iterator i;
   i.it=datos.begin();
   i.final = datos.end();
    if (i.it!=i.final &&suma((*(i.it)).second.root())%2==1)
          ++i;
   return i;
 }
 iterator end(){
   iterator i;
   i.it=datos.end();
   i.final = datos.end();
   return i;
 }
};

int main()
{
   ……..
}
```

Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Pregunta 3

```
#include <iostream>
#include <list>
using namespace std;

template <typename T>   //No es necesaria. Solo está implementada para probar el código en la función main()
void Imprimir(T comienzo,T fin){
  for (auto it =comienzo; it!=fin; ++it){
    cout<<*it<<" ";
  }
}
bool permutalista (list<int> & L1, list<int> & L2){
  if (L1.size()!=L2.size()) return false;
  for (auto it1=L1.begin();it1!=L1.end(); ++it1){
    auto it2=L2.begin();
    bool enc=false;
    while (it2!=L2.end() && !enc){
      if (*it2==*it1){
         it2=L2.erase(it2);
         enc=true;
      }
      else ++it2;
    }
    if (!enc)
      return false;
  }
  return true;
}
int main(){
 list<int>L1={1,23,21,4,2,3,0};    list<int>L2={21,1,3,2,4,23,0};
 cout<<"L1: ";   Imprimir(L1.begin(),L1.end());
 cout<<endl;
 cout<<"L2: ";   Imprimir(L2.begin(),L2.end());
 cout<<endl;
 if (permutalista(L1,L2))
   cout<<"L1 es una premutacion de L2"<<endl;
 else cout<<"L1 no es una permutacion de L2"<<endl;
}
```

Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

Pregunta 4

```
#include <cassert>
#include <list>
#include <iostream>
#include <queue>
#include "bintree.h"
using namespace std;

template <typename T> //No es necesaria. Solo está implementada para probar el código en la función main()
void Imprimir(T comienzo,T fin){
  for (auto it =comienzo; it!=fin; ++it){
    cout<<*it<<" ";
  }
}
void prom_nivel(bintree<int> &T, list<float> &P){
  typedef pair<bintree<int>::node, int> info;
  queue<info> micola;
  int level=0;
  info a(T.root(),0);
  micola.push(a);
  float suma=0;
  int cnt=0;
  while (!micola.empty()){
   a=micola.front();
   micola.pop();
   if (a.second==level){
     suma+=*(a.first);
     cnt++;
   }
   else{       //pasamos a otro nivel
     P.push_back(suma/cnt);
     suma=*(a.first);
     cnt=1;
     level++;
   }
   bintree<int>::node n = a.first;
   info h;
```

```
    if (!n.left().null()){
      h.first = n.left();
      h.second=a.second+1;
      micola.push(h);
    }
    if (!n.right().null()){
      h.first = n.right();
      h.second=a.second+1;
      micola.push(h);
    }
  }
  P.push_back(suma/cnt);
}
int main(){
//
//
//          Ejemplo:                    A
//                                      1
//                                    /   \
//                level 1---->     4      6          V= prom_nivel (A, V)
//                                /   \
//                               8     3              V={ 1, 5 , 5.5 }
//

  bintree<int> a(1);
  a.insert_left(a.root(), 4);
  a.insert_right(a.root(), 6);
  a.insert_left(a.root().right(), 3);
  a.insert_left(a.root().left(), 8);

  list<float>out;
  prom_nivel(a,out);
  cout<<"La lista con los promedios por nivel es..."<<endl;
  Imprimir(out.begin(),out.end());
}
```

UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación e Inteligencia Artificial

Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

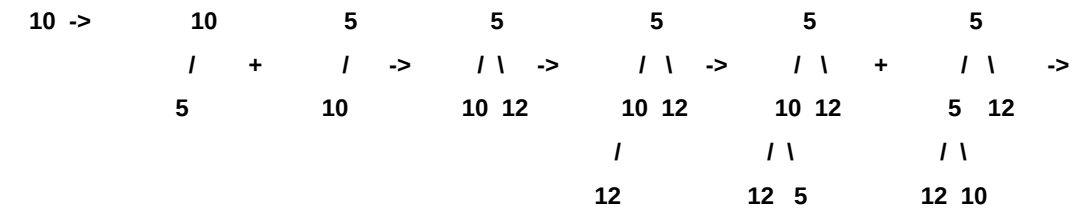
Pregunta 5

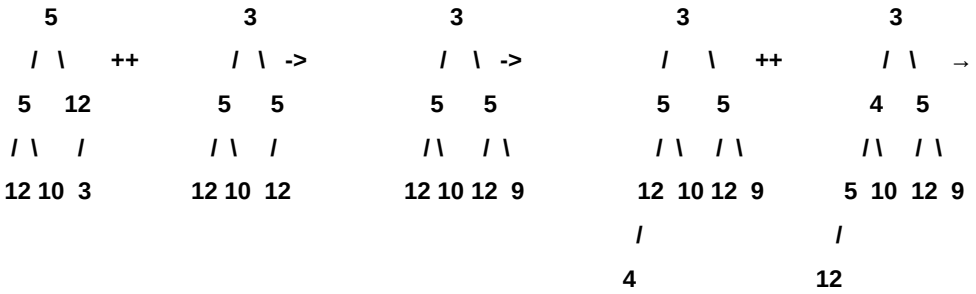
```
#include <iostream>
#include <set>

// Una idea: Si hay un elemento en S cuyo valor es M, fin. Si todos los elementos de S tienen valores mayores que M,  // fin. Si no, probar para cada uno de los elementos x de S con valor x < M para ver si S-{x} tiene un subconjunto // S' con suma M-x. Si esto ocurre, entonces S'+ {x} tiene suma M.


Puede resolverse fácilmente de forma recursiva:

bool tiene_suma_constante (set<int> s, int M){
 if (M==0) return true;
  else{
    if (s.size()==0 ) return false;
    for (auto it = s.begin(); it!=s.end(); ++it){
        if (*it<=M){
           set<int> aux = s;
           aux.erase(*it);
           if (tiene_suma_constante(aux,M-*it)) return true;
                    }
   }
return false;
    }
}

int main(){
  set<int>s={1,2,3,4,5,6};

  if (tiene_suma_constante(s,15)) cout<<"Si existe un subconjunto "<<endl;
  else cout<<"No existe un subconjunto "<<endl;

}
```

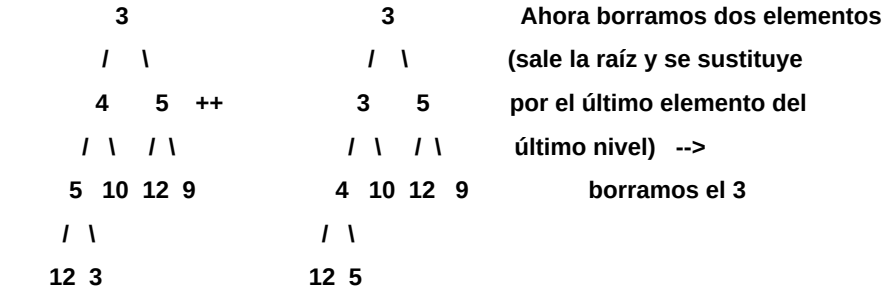
Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

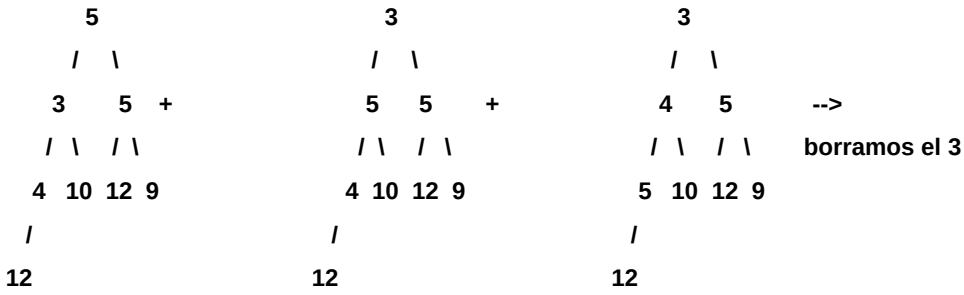
Pregunta 6

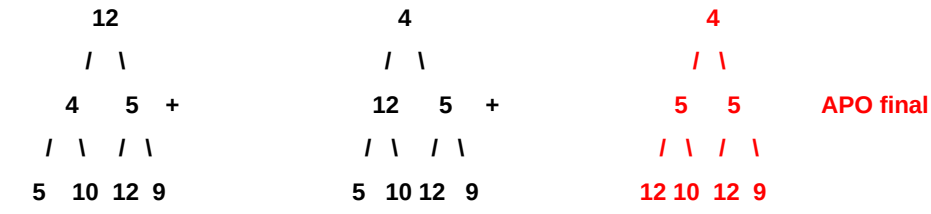
(6a)











Estructuras de datos. Curso 2022-2023
Convocatoria extraordinaria de Febrero.
Grado en Ingeniería Informática.
Doble Grado en Ingeniería Informática y Matemáticas
Doble Grado en Ingeniería Informática y ADE

(6b)

k	47	31	49	66	50	52	82	38	7	63	53
h(k)	8	5	10	1	11	0	4	12	7	11	1
h_o(k)	4	10	6	1	7	9	6	6	8	9	10

h(k) =k%13

h_o(k)=1+k%11

h_i(k)=(h_i-1(k)+h_o(k))%13

Todas se insertan a la primera excepto:

h_2(63)=7

h_3(63)=3

h_2(53)= 11

h_3(53)= 8

h_4(53)= 5

h_5(53)= 2

Tabla resultante

0	1	2	3	4	5	6	7	8	9	10	11	12
52	66	53	63	82	31		7	47		49	50	38