**Juan Ignacio Vera**

***Punto 1(Ejercicio 15):***

**class** **Nodo** {

**int** info;

**Nodo** izquierda, derecha;

**Nodo** (**int** valor)

{ info = valor;

izquierda = **null**;

derecha = **null**; }

**void** **setInfo**(**int** valor)

{ info = valor; }

**void** **setIzquierda**(**Nodo** dir)

{ izquierda = dir; }

**void** **setDerecha**(**Nodo** dir)

{ derecha = dir; }

**int** **getInfo**()

{ **return** info; }

**Nodo** **getIzquierda**()

{ **return** izquierda; }

**Nodo** **getDerecha**()

{ **return** derecha; }

**boolean** **esHoja**()

{

**return** (derecha == **null** && izquierda == **null**);

}

**Nodo** **getUnicoHijo**()

{

**if** ( derecha == **null** ) **return** izquierda;

**else** **return** derecha;

}

**boolean** **tieneUnHijo**()

{

**if** (derecha == **null** && izquierda != **null**)

**return** **true**;

**else** **if** (derecha != **null** && izquierda == **null**)

**return** **true**;

**else**

**return** **false**;

}

**Nodo** **buscaMenor**(**Nodo** p)

{

**Nodo** **aux** = p.getIzquierda();

**while** ( aux.getDerecha() != **null**)

aux = aux.getDerecha();

**return** aux;

}

}

**public** **class** **ArbolBinario**

{

**Nodo** raiz;

**void** **setRaiz**(**Nodo** p)

{**this**.raiz=p;}

**void** **insertar** (**int** valor)

{

**Nodo** **Nuevo**, **P**, **Ant**;

Nuevo = **new** Nodo(valor);

P = raiz;

Ant = **null**;

**while** (P != **null**)

{Ant = P;

**if** (P.getInfo() > valor)

P = P.getIzquierda();

**else** P = P.getDerecha();

}

**if** ( Ant == **null**)

raiz = Nuevo;

**else**

**if** (Ant.getInfo() > valor)

Ant.setIzquierda(Nuevo);

**else** Ant.setDerecha(Nuevo);

}

**void** **eliminar**(**int** valor)

{

**Nodo** **p** = raiz;

**Nodo** **t** = **null**;

**while** ( p.getInfo() != valor )

{ t = p;

**if** ( valor > p.getInfo() )

p = p.getDerecha();

**else**

p = p.getIzquierda();

}

**if** ( p.esHoja() )

**if** ( t == **null** ) raiz = **null**;

**else** **if** ( p == t.getIzquierda() )

t.setIzquierda(**null**);

**else** t.setDerecha(**null**);

**else** **if** (p.tieneUnHijo())

**if** (t.getIzquierda() == p)

t.setIzquierda(p.getUnicoHijo());

**else**

t.setDerecha(p.getUnicoHijo());

**else**

{

**Nodo** **q** = p.buscaMenor(p);

p.setInfo(q.getInfo());

t = p;

p = p.getIzquierda();

**while** (p != q)

{

t = p;

p = p.getDerecha();

}

**if** (q==t.getIzquierda())

t.setIzquierda(q.getIzquierda());

**else**

**if** (q.tieneUnHijo())

t.setDerecha(q.getUnicoHijo());

**else**

t.setDerecha(**null**);

}

}

**void** **inorden** ( **Nodo** P)

{

**if** (P != **null**)

{

inorden (P.getIzquierda());

**System**.***out***.println(P.getInfo());

inorden (P.getDerecha());

}

}

**ArbolBinario** **unirArboles**(**Nodo** raizArbol2)

{

**ArbolBinario** **nuevoArbol**=**new** ArbolBinario();

**Nodo** **Nuevo**, **P**, **Ant**;

Nuevo = raizArbol2;

P = raiz;

Ant = **null**;

nuevoArbol.setRaiz(P);

**while** (P != **null**)

{Ant = P;

**if** (P.getInfo() > raizArbol2.getInfo())

P = P.getIzquierda();

**else** P = P.getDerecha();

}

**if** ( Ant == **null**)

raiz = Nuevo;

**else**

**if** (Ant.getInfo() > raizArbol2.getInfo()) {

Ant.setIzquierda(Nuevo);

**System**.***out***.println(Nuevo.getInfo()+" se inserta a la izquierda de "+ Ant.getInfo());

}**else**{

Ant.setDerecha(Nuevo);

**System**.***out***.println(Nuevo.getInfo()+" se inserta a la derecha de "+ Ant.getInfo());

}

**return** nuevoArbol;

}

}

**public** **class** **MainArbol** {

**public** **static** **void** **main**(**String**[] args)

{

**ArbolBinario** **ab**=**new** ArbolBinario();

**ArbolBinario** **ab2**=**new** ArbolBinario();

**int** **a** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **b** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **c** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **d** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **e** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **f** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **g** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **h** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

ab.insertar(a);

ab.insertar(b);

ab.insertar(c);

ab.insertar(d);

ab.insertar(e);

ab.insertar(f);

ab.insertar(g);

ab.insertar(h);

**int** **i** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **j** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

**int** **k** =(**int**) **Math**.*floor*(**Math**.*random*()\*99+1);

ab2.insertar(i);

ab2.insertar(j);

ab2.insertar(k);

**System**.***out***.println("--------Arbol 1------");

ab.inorden(ab.raiz);

**System**.***out***.println("---------------------");

**System**.***out***.println("--------Arbol 2------");

ab2.inorden(ab2.raiz);

**System**.***out***.println("--------Union 2 arboles------");

**ArbolBinario** **abFinal**=ab.unirArboles(ab2.raiz);

abFinal.inorden(abFinal.raiz);

}

}

***Punto 2:***

**void** **insertar** (**int** valor)

{

**NodoConPunteroPadre** **Nuevo**, **P**, **Ant**;

Nuevo = **new** NodoConPunteroPadre(valor);

P = raiz;

Ant = **null**;

**while** (P != **null**)

{Ant = P;

**if** (P.getInfo() > valor)

P = P.getIzquierda();

**else** P = P.getDerecha();

}

**if** ( Ant == **null**)

raiz = Nuevo;

**else**

**if** (Ant.getInfo() > valor) {

Ant.setIzquierda(Nuevo);

Nuevo.setPadre(Ant);

}**else** {

Ant.setDerecha(Nuevo);

Nuevo.setPadre(Ant);

}

}

**void** **eliminar**(**int** valor)

{

**NodoConPunteroPadre** **p** = raiz;

**NodoConPunteroPadre** **t** = **null**;

**while** ( p.getInfo() != valor )

{ t = p;

**if** ( valor > p.getInfo() )

p = p.getDerecha();

**else**

p = p.getIzquierda();

}

**if** ( p.esHoja() )

**if** ( t == **null** ) raiz = **null**;

**else** **if** ( p == t.getIzquierda() )

t.setIzquierda(**null**);

**else** t.setDerecha(**null**);

**else** **if** (p.tieneUnHijo())

**if** (t.getIzquierda() == p) {

p.getUnicoHijo().setPadre(p.getPadre());

t.setIzquierda(p.getUnicoHijo());

}**else** {

p.getUnicoHijo().setPadre(p.getPadre());

t.setDerecha(p.getUnicoHijo());

}**else**

{

**NodoConPunteroPadre** **q** = p.buscaMenor(p);

p.setInfo(q.getInfo());

t = p;

p = p.getIzquierda();

**while** (p != q)

{

t = p;

p = p.getDerecha();

}

**if** (q==t.getIzquierda()) {

q.getIzquierda().setPadre(q.getPadre());

t.setIzquierda(q.getIzquierda());

}**else**

**if** (q.tieneUnHijo()) {

q.getUnicoHijo().setPadre(q.getPadre());

t.setDerecha(q.getUnicoHijo());

}**else**

t.setDerecha(**null**);

}

}