EXAMEN FEBRERO 2021

TREN

```
del :: Int -> Tren -> Tren
del _ Maquina = error "No se ha encontrado ese peso en el tren"
del w (Vagon cap list tren)
    | elem w list = if newList == [] then tren else (Vagon (cap+w) newList tren)
    | otherwise = (Vagon cap list (del w tren))
        where
            newList = (deleteWeight w list)
            deleteWeight w (x:xs)
                | x == w = xs
                | otherwise = x:(deleteWeight w xs)
```

REVERSE

```
public void reverse() {
        Node<T> current = first;
        Node<T> next;
        Node<T> prev = null;
        while(current != null){
            next = current;
            current = current.next;
            next.next = prev;
            prev = next;
        }
        first = prev;
    }
```

TRIM

```
trim :: BinTree a -> BinTree a
trim (Node x Empty Empty) = Empty
trim (Node x tree1 Empty) = (Node x (trim tree1) Empty)
trim (Node x Empty tree2) = (Node x Empty (trim tree2))
trim (Node x tree1 tree2) = (Node x (trim tree1) (trim tree2))
```

```
public void trim() {
        if(root != null)
            this.root = trimRec(root);
```

```
    }

    private Tree<T> trimRec(Tree<T> node){
        if ((node.left == null) && (node.right == null)){
            return null;
        } else if (node.left == null){
            node.right = trimRec(node.right);
        } else if (node.right == null){
            node.left = trimRec(node.left);
        } else {
            node.left = trimRec(node.left);
            node.right = trimRec(node.right);
        }

        return node;

    }
```

ISOMORPHIC

```
isomorphic :: BinTree a -> BinTree b -> Bool
isomorphic Empty Empty = True
isomorphic Empty (Node x lt rt) = False
isomorphic (Node x lt rt) Empty = False
isomorphic (Node x lt1 rt1) (Node y lt2 rt2) = isomorphic lt1 lt2 && isomorphic rt1 rt2
```

```
private static <A, B> boolean isomorphicRec(Tree<A> t1, Tree<B> t2) {
        if(t1 == null && t2 == null){
            return true;
        }else if(t1 == null && t2 != null){
            return false;
        }else if(t1 != null && t2 == null){
            return false;
        }else{
            return isomorphicRec(t1.left, t2.left) && isomorphicRec(t1.right, t2.right);
        }
    }
```