Exercise 2.65.

Use the results of exercises 2.63 and 2.64 to give $\Theta(n)$ implementations of `union-set` and `intersection-set` for sets implemented as (balanced) binary trees.[1]

Answer.

It is rather natural to perform union and intersection operations for (balanced) binary trees by first converting them into ordered lists, then do the operations on these lists and finally transform these lists back into (balanced) binary trees.

On the other hand, we have learnt from exercise 2.63 that two procedures of converting a binary tree to a list `tree->list-1` and `tree->list-2` can be done in $\Theta(n \log n)$ steps and $\Theta(n)$ steps respectively. We also discovered in exercise 2.64 that the procedure `list->tree` which transforms a list to a tree requires $\Theta(n)$ steps. Additionally, remember that when we represent set as ordered lists in the former text, both the `union-set` and `intersection-set` operations take up $\Theta(n)$ number of steps. Hence, we can implement the required $\Theta(n)$ `union-set` and `intersection-set` on sets represented as (balanced) binary trees by combining the `tree->list-2`, `list->tree` operations together with the corresponding set operations on ordered lists.[2]

```
(define (union-set set1 set2)
  (list->tree
   (union-list (tree->list-2 set1)
               (tree->list-2 set2))))

(define (intersection-set set1 set2)
  (list->tree
   (intersection-list (tree->list-2 set1)
                      (tree->list-2 set2))))
```

*. Creative Commons 2013, Lawrence R. Amlord(颜世敏).

1. Exercise 2.63–2.65 are due to Paul Hilfinger.

2. To avoid confusing operations on sets represented as binary trees with those of ordered lists, we rename the corredponding operations on sets represented as ordered lists as `union-list` and `intersection-list`. Their implementations both have been addressed in the former text and exercise 2.59 respectively.