Exercise 2.60.

We specified that a set would be represented as a list with no duplicates. Now suppose we allow duplicates. For instance, the set $\{1, 2, 3\}$ could be represented as the list `(2 3 2 1 3 2 2)`. Design procedures `element-of-set?`, `adjoin-set`, `union-set`, and `intersection-set` that operate on this representation. How does the efficiency of each compare with the corresponding procedure for the non-duplicate representation? Are there applications for which you would use this representation in preference to the non-duplicate one?

Answer.

In this representation, `element-of-set?` is pretty the same as the non-duplicate one:

```
(define (element-of-set? x set)
  (cond ((null? set) false)
        ((equal? x (car set)) true)
        (else (element-of-set? x (cdr set)))))
```

Using this representation, we can implement `adjoin-set` by simply adding the element at the head of an existing set:

```
(define (adjoin-set x set)
  (cons x set))
```

Just as `element-of-set?`, the `intersection-set` operation remain the same even though we allow duplicates:

```
(define (intersection-set set1 set2)
  (cond ((or (null? set1) (null? set2)) '())
        ((element-of-set? (car set1) set2)
         (cons (car set1)
               (intersection-set (cdr set1) set2)))
        (else (intersection-set (cdr set1) set2))))
```

Things become easy when we perform the `union-set` operation in this duplicate representation, all we have to do is to roughly combine two existing set into one:

```
(define (union-set set1 set2)
  (append set1 set2))
```

Now let's come to investigate the efficiency of these operations. Notice that both the predicate the `element-of-set?` and the `intersection-set` operation remain the same no matter whether the set allow duplicate or not. Hence, their efficiencies stay invariant with an order of growth as $\Theta(n)$ and $\Theta(n^2)$ respectfully. The `adjoin-set` operation, obviously, can be accomplish by one step—just add that particular element in the front of a designated set. Thus, the number of steps required grows as $\Theta(1)$. Since the operation `union-set` relies on another operation `append`, which has an order of growth of $\Theta(n)$. Hence, the number of steps it required also grows as $\Theta(n)$.

Although, this duplicate representation might seem awkward to us, there are still many application in peference to it. For example, when we try to store the raw data on statistics, representation of set that allow duplicate will therefore becomes a better choice.