Exercise 2.45.

`Right-split` and `up-split` can be expressed as instances of a general splitting operation. Define a procedure `split` with the property that evaluating

```
(define right-split (split beside below))
(define up-split (split below beside))
```

Answer.

We've seen that `right-split` and `up-split` each makes the painters split and branch in a square pattern; they differ only in how they orient their half and quarter copies. Hence, we can abstract this pattern of painter combination with the `split` procedure, which takes two dual-argument painter operations and produce a painter operation that splits and branches a given painter with those two operations to a designated depth. `Half` and `quarter` are the splitting and branching operations to apply to the half copy and quarter copy respectively.

```
(define (split half quarter)
  (lambda (painter n)
    (if (= n 0)
        painter
        (let ((smaller ((split half quarter) painter (- n 1))))
          (half painter (quarter smaller smaller))))))
```