Question 1

Consider the following data definitions for `BinaryTree` and `Path`

```
(define-struct node (k v l r))
;; BinaryTree is one of:
;; - false
;; - (make-node Natural String BinaryTree BinaryTree)
;; interp. a binary tree, each node has key, value, and l/r children
(define BT0 false)
(define BT1 (make-node 1 "a" false false))
(define BT4 (make-node 4 "d"
                       (make-node 2 "b")
                                  (make-node 1 "a" false false)
                                  (make-node 3 "c" false false))
                       (make-node 5 "e" false false)))
;; Path is one of:
;; - empty
;; (cons "L" Path)
;; (cons "R" Path)
;; interp. a sequence of left and right 'turns' down though a BinaryTree
;;         (list "L" "R" "R" means take the left child of the root, then
;;         the right child of that node, and the right child again.
;;         empty means you have arrived at the destination.
(define P1 empty)
(define P2 (list "L")
(define P3 (list "R")
(define P4 (list "L" "R")
```

Question 1

1/1 point (graded)

We want to write a function that consumes a BinaryTree and a Path.

How many cells will the resulting cross-product table have?

- 2, a 2x1 table
- 4, a 2x2 table
- [✓ 6, a 2x3 table]
- 9, a 3x3 table

✓

Explanation

BinaryTree has 2 cases, and Path has 3 cases, so we will get a 2x3 table with 6 cells.

Submit

Answers are displayed within the problem