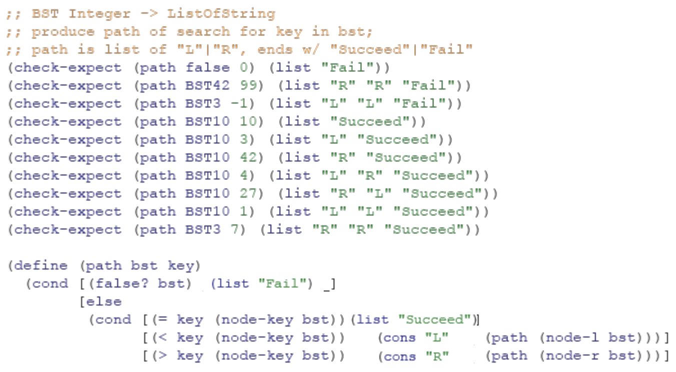
Question 1

Question 1

1/1 point (graded)
Suppose instead of producing the value of a node with a given key, we would like to produce the path that we took when we searched for that key. So the function that we would like to design should produce a list of "L" or "R", and should end with either "Fail" or "Succeed", depending on whether we found the key.

We're almost done designing this function. We have the signature, purpose, tests, and we've even started filling in the template.

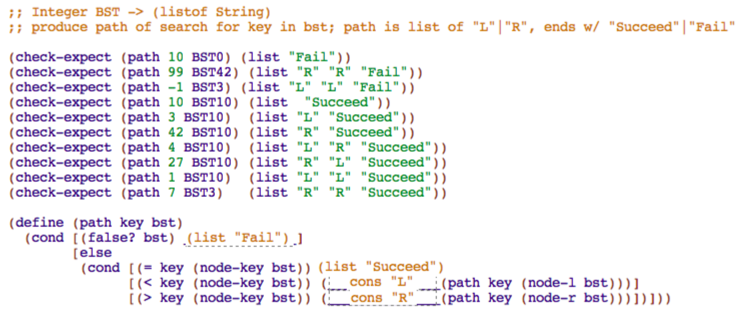
Drag the expressions below, into the appropriate box on the partially filled in template.



```
;; BST Integer -> ListOfString
;; produce path of search for key in bst;
;; path is list of "L"|"R", ends w/ "Succeed"|"Fail"
(check-expect (path false 0) (list "Fail"))
(check-expect (path BST42 99) (list "R" "R" "Fail"))
(check-expect (path BST3 -1) (list "L" "L" "Fail"))
(check-expect (path BST10 10) (list "Succeed"))
(check-expect (path BST10 3) (list "L" "Succeed"))
(check-expect (path BST10 42) (list "R" "Succeed"))
(check-expect (path BST10 4) (list "L" "R" "Succeed"))
(check-expect (path BST10 27) (list "R" "L" "Succeed"))
(check-expect (path BST10 1) (list "L" "L" "Succeed"))
(check-expect (path BST3 7) (list "R" "R" "Succeed"))

(define (path bst key)
  (cond [(false? bst)  (list "Fail") _]
        [else
         (cond [(= key (node-key bst))(list "Succeed")]
               [(< key (node-key bst))   (cons "L"    (path (node-l bst)))]
               [(> key (node-key bst))   (cons "R"    (path (node-r bst)))]
```

✓



```
;; Integer BST -> (listof String)
;; produce path of search for key in bst; path is list of "L"|"R", ends w/ "Succeed"|"Fail"

(check-expect (path 10 BST0) (list "Fail"))
(check-expect (path 99 BST42) (list "R" "R" "Fail"))
(check-expect (path -1 BST3) (list "L" "L" "Fail"))
(check-expect (path 10 BST10) (list  "Succeed"))
(check-expect (path 3 BST10)  (list "L" "Succeed"))
(check-expect (path 42 BST10) (list "R" "Succeed"))
(check-expect (path 4 BST10)  (list "L" "R" "Succeed"))
(check-expect (path 27 BST10) (list "R" "L" "Succeed"))
(check-expect (path 1 BST10)  (list "L" "L" "Succeed"))
(check-expect (path 7 BST3)   (list "R" "R" "Succeed"))

(define (path key bst)
  (cond [(false? bst) (list "Fail") ]
        [else
         (cond [(= key (node-key bst)) (list "Succeed")
               [(< key (node-key bst)) (  cons "L"  (path key (node-l bst)))]
               [(> key (node-key bst)) (  cons "R"  (path key (node-r bst)))])]))
```

Submit

Answers are displayed within the problem