Tree

Friday, January 13, 2023 2:52 PM

```
def in_order_traversal(t):
    """
    Generator function that generates an "in-order" traversal, in which we
    yield the value of every node in order from left to right, assuming that each node has either 0 or 2
branches.
    For example, take the following tree t:
           1
       2       3
    4     5
        6  7
    We have the in-order-traversal 4, 2, 6, 5, 7, 1, 3
    >>> t = Tree(1, [Tree(2, [Tree(4), Tree(5, [Tree(6), Tree(7)])]), Tree(3)])
    >>> list(in_order_traversal(t))
    [4, 2, 6, 5, 7, 1, 3]
    """
    "*** YOUR CODE HERE ***"
    ans = []
    if t.is_leaf():
        ans.append(t.label)
        return ans #return the label of leaves
    ans.extend(in_order_traversal(t.branches[0])) #selector that return left child node
    ans.append(t.label)#parent
    ans.extend(in_order_traversal(t.branches[1]))#sibling
    return ans
```

Scheme

Friday, January 13, 2023 3:42 PM

```scheme
(define (reverse lst)
    (if (null? lst) nil
    (append (reverse (cdr lst)) (list (car lst))))
)
;ban be revealed to quasitation
```

Recursion

Friday, January 13, 2023 4:51 PM

```
def interleaved_sum(n, odd_term, even_term):
    """Compute the sum odd_term(1) + even_term(2) + odd_term(3) + ..., up
    to n.
    >>> # 1 + 2^2 + 3 + 4^2 + 5
    ... interleaved_sum(5, lambda x: x, lambda x: x*x)
    29
    >>> from construct_check import check
    >>> check(SOURCE_FILE, 'interleaved_sum', ['While', 'For', 'Mod']) # ban
loops and %
    True
    """
    "*** YOUR CODE HERE ***"
    def helper(k, flag):
        if k > n:
            return 0
        if flag == 1:
            return even_term(k) + helper(k + 1, flag -1)
        else:
            return odd_term(k) + helper(k + 1, flag +1)
    return helper(0, 1)

    #another sulotion for this kind question but been banned for this one
since no %
    # result = 0
    # def helper(n, odd_term, even_term):
    #     nonlocal result
    #     if n == 0:
    #         return result
    #     elif n % 2 == 0:
    #         result = result + even_term(n)
    #         return helper(n-1, odd_term, even_term)
    #     else:
    #         result = result + odd_term(n)
    #         return helper(n-1, odd_term, even_term)
    # return helper(n, odd_term, even_term)
```

Mutate

Friday, January 13, 2023 5:27 PM

```
def mutate_reverse(link):
    """Mutates the Link so that its elements are reversed.
    >>> link = Link(1)
    >>> mutate_reverse(link)
    >>> link
    Link(1)
    >>> link = Link(1, Link(2, Link(3)))
    >>> mutate_reverse(link)
    >>> link
    Link(3, Link(2, Link(1)))
    """
    "*** YOUR CODE HERE ***"
    #notice scheme tail recursion reverse is not the same as in python
    value = [] #use a stack to store, an intermedia
    ptr = link
    while ptr is not Link.empty:
        value.append(ptr.first)
        ptr = ptr.rest
        #transfer all into value
    while link is not Link.empty:
        link.first = value.pop() #pop means pop the last element in stack
        link = link.rest
```