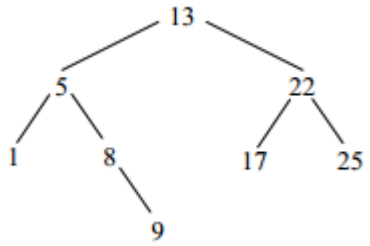
Problem 1. Consider an implementation of binary trees with Scheme lists, as in the following example:

```
(define T
  '(13
     (5
        (1 () ())
        (8 ()
           (9 () ())))
(22
   (17 () ())
   (25 () ()))))
```



(a) Write a recursive function (n-nodes T), which returns the number of nodes in the tree T. The following example illustrates the use of this function:

```
> (n-nodes T)
8
```

(b) Write a recursive function (n-leaves T), which returns the number of leaves in the tree T. The following example illustrates the use of this function:

```
> (n-leaves T)
4
```

(c) The height of a tree is defined as the maximum number of nodes on a path from the root to a leaf. Write a recursive function (height T), which returns the height of the tree T. The following example illustrates the use of this function:

```
> (height T)
4
```

(d) Write a recursive function (postorder T), which returns the list of all elements in the tree T corresponding to a postorder traversal of the tree. The following example illustrates the use of this function:

> (postorder T)

(1 9 8 5 17 25 22 13)

Problem 2. A binary search tree is a binary tree for which the value in each node is greater than or equal to all values in its left subtree, and less than all values in its right subtree. The binary tree given as example in problem 3 also qualifies as a binary search tree.

Using the same list representation, write a recursive function (member-bst? V T), which determines whether V appears as an element in the binary search tree T. The following example illustrates the use of this function:

> (member-bst? 17 T)

#t

Solution:

```
; Problem 1

(define T
'(13
(5
(1 () ())
(8 ()
(9 () ()))))
(22
(17 () ())
(25 () ()))))

; Define three auxiliary functions (left T),(right T) and (val T) which
return the left subtree,
; the right subtree, and the value in the root of tree T, respectively.

(define (left L)
   (list-ref L 1)
)

(define (right L)
   (list-ref L 2)
)
```

```
(define (val L)
   (list-ref L 0)
)

(define (n-nodes L)
   (cond
      [(equal? (length L) 0) 0]
      [else (+ 1 (+ (n-nodes (left L)) (n-nodes (right L))))]
    )
)

(define (n-leaves L)
(cond [(equal? (length L) 0) 0]
[else
   (cond
      [(equal? (+ (length (left L)) (length (right L))) 0) 1]
      [else (+ (n-leaves (left L)) (n-leaves (right L)))]
    )
]
)
)

(define (height L)
   (cond [(equal? (length L) 0) 0]
         [else (+ 1 (max (height (left L)) (height (right L))))]
   )
)

(define (postorder L)
   (cond [(equal? (length L) 0) ()]
   [else (append (postorder (left L)) (postorder (right L))
          (list (val L)))
   ]
   )
)

; Problem 2

(define (member-bst? x L)
   (cond [(equal? (length L) 0) #f]
         [else
         (cond [(equal? x (val L)) #t]
               [else
               (cond [(> x (val L)) (member-bst? x (right L))]
                     [else (member-bst? x (left L))]
               )
               ]
         )
         ]
    )
)
```