CS326 – Programming Languages, Concepts and Implementation

Homework 2

(Due February 23)

Submission:

- Submit your code in Canvas as one "hw2.txt" file containing all your functions. You may generate this file with any plain text editor.
- The file must be able to load and be tested in the interpreter.
  - Consequently, the file must be in a plain text format; do not submit Word, PDF, RTF, JPG or any such types of files.
  - Also make sure that any auxiliary information (such as your name or question numbers) is commented out.

1. (13 pts) Write a recursive Scheme function (`subst x y L`), which returns a list identical to L except that every occurrence of `x` has been replaced with `y`. The following example illustrates the use of this function:

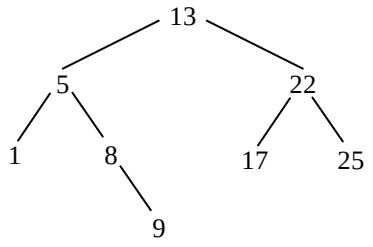
```
> (subst 'c 'k '(c o c o n u t))
(k o k o n u t)
```

2. (13 pts) Write a recursive Scheme function (`all-different? L`), which determines whether all elements of list L are distinct (that is, not `equal?`). The following example illustrates the use of this function:

```
> (all-different? '(3 7 2 7))
#f
```

3. (60 pts) Consider an implementation of binary trees with Scheme lists, as in the following example:

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



Before proceeding, it may be useful to 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.

(a) (15 pts) 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) (15 pts) 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) (15 pts) 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) (15 pts) 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)
```

4. (14 pts) Write a recursive Scheme function `(flatten L)`, which takes as arguments a list `L` (possibly containing sublists), and returns a list containing all elements in `L` and its sublists, but all at the same level. The following example illustrates the use of this function:

```
> (flatten '(1 (2 (3 4)) 5))
(1 2 3 4 5)
```

5. (Extra Credit - 10 pts) 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`. Make sure you actually use binary search. The following example illustrates the use of this function:

```
> (member-bst? 17 T)
#t
```