Tutorial Problem Set 4

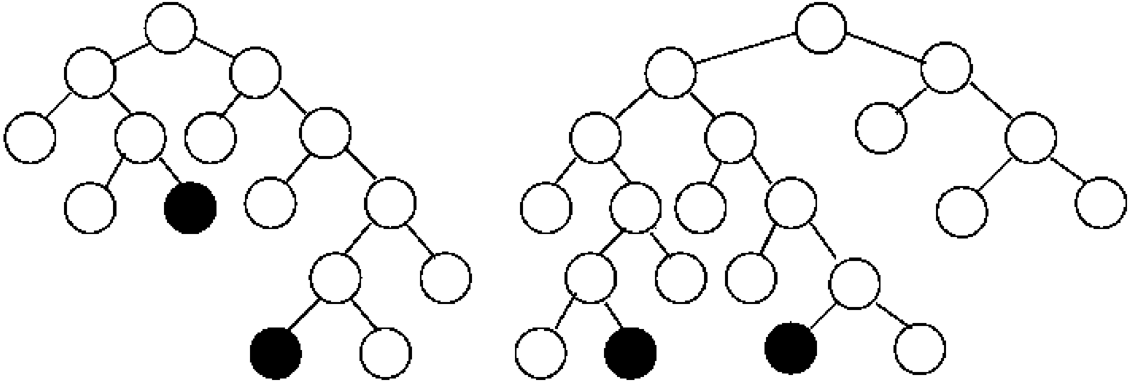
CS 152 – Abstractions and Paradigms in Programming

Date - 18th Feb 2017

1. Consider a tree defined by the following structs:

   ```
   (struct node (ltree val rtree) #:transparent)
   (struct leaf(val) #:transparent)
   ```

   The diameter of a tree is the length of the longest path from one leaf to another. For this question, the length of a path is a count of all the interior nodes in the path and also includes both the leaves at the ends. Write a function (dia t) that will return the diameter of the tree t. The figure below illustrates the notion of diameter. For each of the two trees, the diameter is 9 and the longest path is between the shaded leaves.



2. Write a procedure (list-within bst lb ub) that takes as arguments a binary search tree of numbers, a lower bound and an upper bound, and returns an ordered list of values at the nodes that are greater than or equal to the lower bound and less than or equal to the upper bound. Assume that the tree is a binary search tree, possibly with duplicate values (duplicate values are shared only between a root and its left subtree). For full credit, your function should run in $O(n)$ time and should not examine more of the tree than is necessary. n is the number of nodes in the tree.

   Use the structs:

   ```
   (struct node (value ltree rtree) #:transparent)
   (struct nulltree () #:transparent)
   ```

   Test your program for the following tree:

```
(define t1
 (node 10
    (node 10
       (node 5 (nulltree) (nulltree))
       (nulltree))
    (node 15 (nulltree) (nulltree))))

 (list-within t1 1 100) => (5 10 10 15)
 (list-within t1 5 10)  => (5 10 10)
 (list-within t1 10 10) => (10 10)
 (list-within t1 5 5)   => (5)
```

3. Let us call a binary tree symmetric if its right subtree is the mirror image of its left subtree. Write a function `(all-sym-trees n)` which will generate a list of all symmetric trees which have exactly `n` interior nodes. Assume that `n` is odd.

   In this question, we shall only be interested in the structure of the tree. Therefore use the following `structs` to represent your trees.

   In this question, we shall only be interested in the structure of the tree. Therefore use the following `structs` to represent your trees.

   ```
   (struct node (ltree rtree) #:transparent)
   (struct leaf () #:transparent)
   ```

4. Consider a general tree defined by the following struct:

   ```
   (struct gnode (val  lst) #:transparent)
   ```

   A node of a general tree is at level n if the path from the root to the node has length $n-1$. The root node is at level 1. Write a function `(atlevel t n)` to collect all nodes at a given level n in a tree t. If n is greater than the height of the tree, the `atlevel` returns the null list.

5. We want to represent UNIX directories in the form of trees. For this purpose, we have defined the following structures:

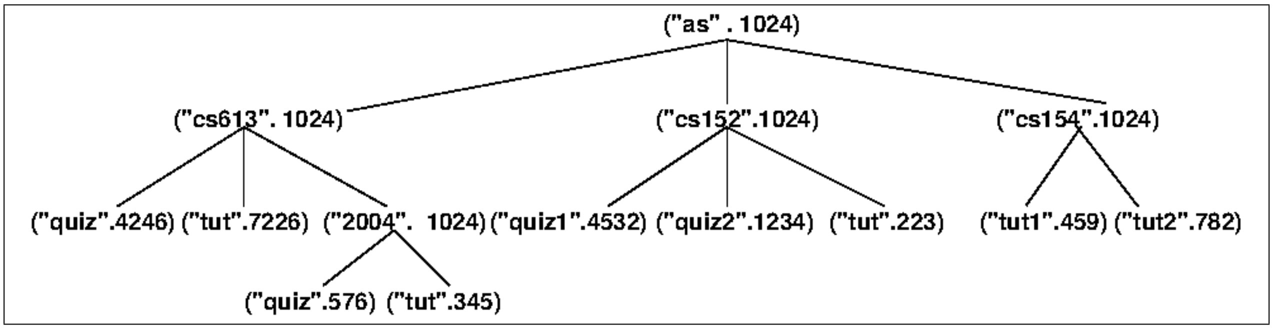
   ```
   (struct dir (info fdlist) #:transparent)
   (struct file (info contents) #:transparent)
   ```

   Here

   (a) `info` is a cons-ed pair `(name.size)`, where `name` is a string representing the name of the file or the directory and `size` is a `n` integer representing its size.

   (b) `fdlist` is a list of files and directories and contents is a string representing the contents of the file.

(c) pathname is a list of strings. For example, ("as" "cs613" "2004") is an example of a pathname.



(a) Write function findtree which takes a directory tree and a pathname and returns the subdirectory represented by the pathname.

(b) Define a function ls which takes a directory and a pathname an d lists the names of all the files and directories in the sub-directory represented by the pathname. The files under a directory should be listed immediately after the directory is listed.

(c) Define a function called size which takes a directory tree and a pathname, and returns the size of the entire directory tree represented by the pathname.

(d) Define a function called delete which takes a directory tree and a pathname and returns the tree with the subdirectory or the file represented by the pathname deleted.

To keep things simple, assume that the file or the directory represented by the pathname exists in the directory tree. You can use the following example:

```
(define thistree
  (dir
   (cons "as" 1024)
   (list (dir
          (cons "cs613" 1024)
          (list (file (cons "quiz" 4246) "junk")
                (file (cons "tut" 7226) "junk")
                (dir (cons "2004" 1024)
                     (list (file (cons "quiz" 576) "junk")
                           (file (cons "tut" 345) "junk")))))
         (dir (cons "cs152" 1024)
              (list (file (cons "quiz1" 4532) "junk")
                    (file (cons "quiz2" 1234) "junk")
                    (file (cons "tut" 1223) "junk")))
         (dir (cons "cs154" 1024)
              (list (file (cons "tut1" 459) "junk")
                    (file (cons "tut2" 782) "junk"))))))
```