

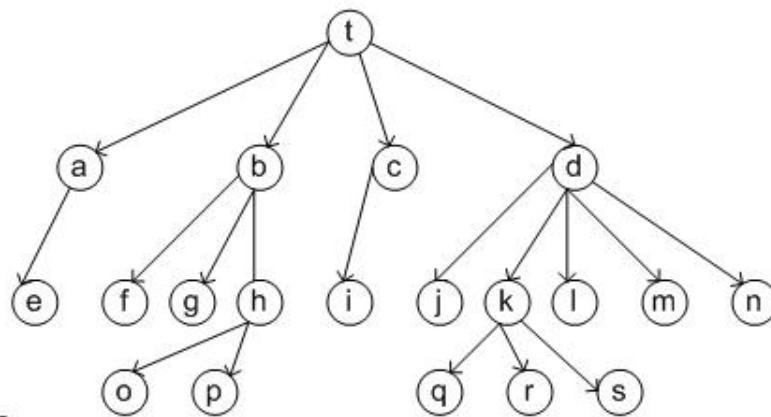
CS2420 – Fall 2008
Program 1
Due September 9, 2008

This assignment assumes that you have written binary tree routines. If it has been several years since you took CS 1410, you need to get started immediately.

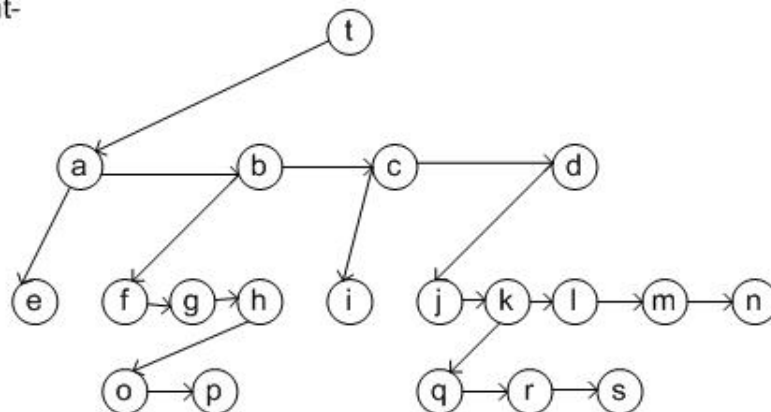
A general tree is a tree in which the number of children is not restricted to two. The problem with a general tree is that if we use an array to store the children pointers, we need to know a maximum number of children.

A more flexible solution (and one that requires less pointer space) stores the tree as a “left most child” and “next right sibling.” Instead of storing pointers to all children, a node only stores a pointer to its leftmost child. That child stores a pointer to its next right sibling. See the example below.

General Tree



General Tree stored as left-most-child/next-right-sibling



For this assignment, you are to complete the following:

1. Create the above tree from a preorder traversal in which each node knows the number of children it has. The input (prog1.txt) looks like where the letter is the name of the node and the number is the number of children:

```

t 4
a 1
e 0
b 3
f 0
g 0
h 2
o 0
p 0
c 1
i 0
d 5
j 0
k 3
q 0
r 0
s 0
l 0
m 0
n 0

```

- Print out the tree prettily, in preorder, with each node on a line by itself and indented to show level. The above tree would look like:

```

t
  a
    e
  b
    f
    g
    h
      o
      p
  c
    i
  d
    j
    k
      q
      r
      s
    l
    m
    n

```

- Write the code to find the height of the tree (counting the root as level 1).
- Write the code to print the level that has the maximum number of nodes (counting the root as level 1).
- Write the code to create the mirror image of the tree (the first child becomes the last child, recursively). Print the reversed tree.

Hint

It is helpful to have a method that takes the input file as a parameter and returns the tree node made by creating the next **subtree** the input file contains.

```
TreeNode * Tree:: makeTree(ifstream & fin)
```

Creating a mirror image is a bit tricky. Draw it out so you can see clearly what needs to happen. I thought of the problem as disconnecting a child from its parent and then reconnecting it to another list of children. When you disconnect, be careful to change rightSib pointers. As I change the rightSib pointer, I had to make sure I didn't lose the sibling.

I used something like:

```
for ( TreeNode * kid = t->leftChild; kid != NULL; kid = next )
{  next = kid->rightSib;
   kid->rightSib = . . .
   // more good stuff
} // for
```

Turn In

When complete submit a .zip file (and only .zip files, not .tar, not .rar, not .7z, just a .zip file) containing your entire project solution directory. Remember to delete the \Debug and \Release directories, along with the .ncb file.