Small Group Tutorial 2, 27/2 - 3/3/2017
Solutions

1. Let T be a *ordered tree* with more than one node. Is it possible that the *preorder traversal* of T visits the nodes *in the same order* as the *postorder traversal* of T? If so, give an example; otherwise, argue why it cannot occur.

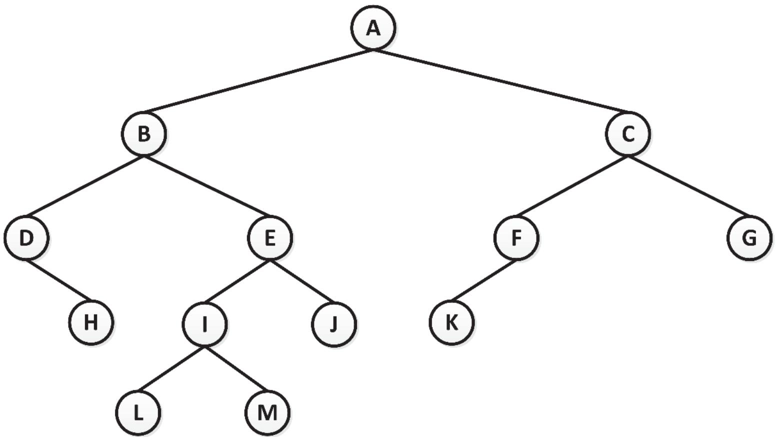
   Is it possible that the preorder traversal of T visits the nodes in *the reverse order* of the postorder traversal of T? If so, give an example; otherwise, argue why it cannot occur.

   Answer

   It is not possible for the postorder and preorder traversal of a tree with more than one node to visit the nodes in the same order. A preorder traversal will always visit the root node first, while a postorder traversal node will always visit an external node first.

   It is possible for a preorder and a postorder traversal to visit the nodes in the reverse order. Consider the case of a tree with only two nodes.

2. Let T be the binary tree as below



(a) Give the output of toStringPostorder(T,T.root()) method presented below.

```
public static String toStringPostorder(Tree T, Position v){
        String s = "";
        for (Position w: T.children(v))
            s += toStringPostorder(T, w) + ", ";
        s += v.element().toString();        // main visit action
        return s;
}
```

(b) Give the output of toStringPreorder(T,T.root()) method presented below.

```
public static String toStringPreorder(Tree T, Position v){
        String s =  s += v.element().toString();        // main visit action
            ;
        for (Position w: T.children(v))
            s += ", " + toStringPreorder(T, w);

        return s;
}
```

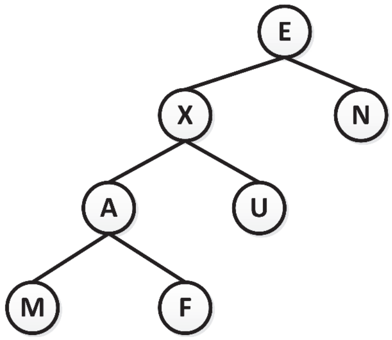
<u>Answer</u>

(a) s="H, D, L, M, I, J, E, B, K, F, G, C, A"

(b) s= "A, B, D, H, E, I, L, M, J, C, F, K, G"

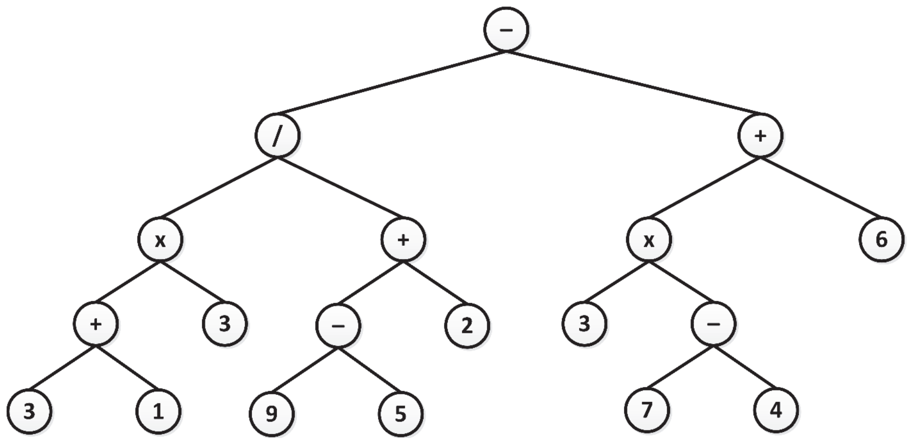
3. Draw a (single) binary tree T such that:
   - Each internal node of T stores a single character
   - A preorder traversal of T yields E X A M F U N
   - An inorder traversal of T yields M A F X U E N

Answer



4. Let T be the binary tree as below.
   Give the output of printExpression(T,T.root()) algorithm presented below.

```
Algorithm: printExpression(Tree T, Position v)
    if T.isInternal(v) then
        print "("
    if T.hasLeft(v) then
        printExpression(T, T.left(v))
    if T.isInternal(v) then
        print the operator stored at v
    else
        print the value stored at v
    if T.hasRight(v) then
        printExpression(T, T.right(v))
    if T.isInternal(v) then
        print ")"
```



<u>Answer</u>

$((((3+1)\text{x}3)/((9-5)+2)) - ((3\text{x}(7-4))+6))$

5. Proof the following properties of binary trees (for $n \geq 1$)

   (a) $1 \leq n_e \leq 2^h$

   (b) $h \leq n_i \leq 2^h - 1$

   (c) $\log_2(n+1) - 1 \leq h \leq n - 1$

   where n – number of nodes; n_e – number of external nodes; n_i – number of internal nodes; h – height of a binary tree

   Answer

   (a) $1 \leq n_e \leq 2^h$
   Justification:
   Binary tree with only one node (root) has exactly one external node.
   Binary tree with the height h can have up to 2^h external nodes if it is a proper binary tree. Note that each parent has only two children (left and right).

   (b) $h \leq n_i \leq 2^h - 1$
   Justification:
   Binary tree with only one node (root) has no internal nodes and its height is $h = 0$, thus $h \leq n_i$.
   The proper binary tree with the height h has the following number of nodes n:

$$n = \sum_{i=0}^{h} 2^i = 1 + 2 + 4 + \cdots + 2^h$$

   Using the formula for a geometric series:

$$\sum_{k=0}^{x} a \cdot r^k = \frac{a \cdot (1 - r^{n+1})}{1 - r}$$

   for $a = 1$, $r = 2$, and $x = h$ we obtain:

$$n = \sum_{i=0}^{h} 2^i = 1 + 2 + 4 + \cdots + 2^h = \frac{1 \cdot (1 - 2^{h+1})}{1 - 2} = 2^{h+1} - 1$$

   Number of internal nodes can be calculated as:

$$n_i = n - n_e$$

   From point (a) we know that the maximum number of n_e in a binary tree is 2^h so the maximum number of n_i is:

$$n_i = n - n_e$$
$$n_i = 2^{h+1} - 1 - 2^h$$
$$n_i = 2^h - 1$$

(c) $\log_2(n+1) - 1 \le h \le n - 1$
Justification:
During the lecture the proof for the following was presented:

$$h + 1 \le n \le 2^{h+1} - 1$$

This can be used to infer that $\log_2(n+1) - 1 \le h \le n - 1$ in the following way:

$$h + 1 \le n \le 2^{h+1} - 1$$

$$h \le n - 1 \text{ and } n + 1 \le 2^{h+1}$$

$$h \le n - 1 \text{ and } \log_2(n+1) \le h + 1$$

$$\log_2(n+1) - 1 \le h \le n - 1$$

6. ADDITIONAL – Describe in pseudo–code, a nonrecursive method for performing an inorder traversal of a binary tree T.

Answer

```
Algorithm inorder(Tree T):
    Stack S ← new Stack()
    Node v ← T.root()
    push v
    while S is not empty do
        while v is internal do
            v ← v.left
            push v
        while S is not empty do
            pop v
            visit v
            if v is internal then
                v ← v.right
                push v
            while v is internal do
                v ← v.left
                push v
```