Algorithms & Computation Lab Sheet 4A Binary Trees

1. Take and modify your code from Lab Sheet 3B (Binary Tree) for a binary search tree of Chars, and code the other two traversals (Pre and Post Order).

2. Take your code from Lab Sheet 3B (Binary Tree) for a binary tree of integers, and code a method called hasPathSum() which given a binary tree and a sum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals the given sum. Return false if no such path can be found. The function prototype is `int hasPathSum(struct node* node, int sum)`

   Note: a "root-to-leaf path" is a sequence of nodes in a tree starting with the root node and proceeding downward to a leaf (a node with no children). An empty tree contains no root-to-leaf paths. So for example, the following tree has exactly four root-to-leaf paths:

   ```
        5
       / \
      4   8
     /   / \
    11  13  4
   / \       \
  7   2       1
   ```

   Root-to-leaf paths:
   path 1: 5 4 11 7
   path 2: 5 4 11 2
   path 3: 5 8 13
   path 4: 5 8 4 1

   For this problem, we will be concerned with the sum of the values of such a path -- for example, the sum of the values on the 5-4-11-7 path is 5 + 4 + 11 + 7 = 27.

3. Now code a method called printPaths() which prints out all of its root-to-leaf paths as defined above. The function prototype is `void printPaths(struct node* node)`

   In this problem the "path so far" needs to be communicated between the recursive calls. To get over this problem you could create a recursive helper function `printPathsRecur(node, int path[], int pathLen)`, where the path array communicates the sequence of nodes that led up to the current call.