CMSC 123: Data Structures

Handout Adapted from: Clinton Poserio

Exercise 04: BST::Traversals (In-Lab Exercise)

Overview

Generally, trees have traversal operations. A traversal or a *tree walk* is a process where in each node in the tree is visited. Since, a tree is a non-linear structure, there is no unique traversal. For `BST` ADT, we consider the following traversal algorithms:

1. Inorder tree walk
2. Preorder tree walk
3. Postorder tree walk

These traversal algorithms are simple recursive algorithms that visits each node in a BST, beginning from the root, in a unique order. Using ***inorder tree walk*** allows us to print the keys in a BST in an increasing order (*i.e.* a sorted order). As the name suggests, this traversal visits (and prints) the root node of a subtree *in between* its left subtree and right subtree; that is, for each node you visit, visit first its left child, then the node, and finally, the right child. The first node to be visited is the root.

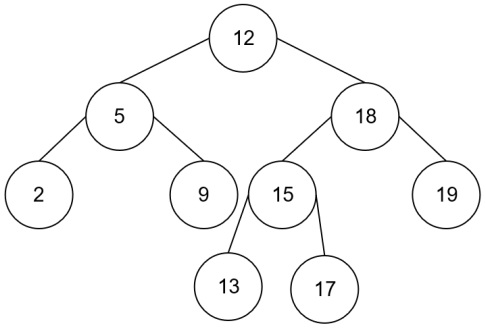

Figure 1: Sample BST [*rooted at 12*]

Applying the algorithm to the BST in Figure 1, we have the following:

```
visit node 12
  visit left of 12: node 5
    visit left of 5: node 2
      visit left of 2 (NULL)
      print 2
      visit right of 2 (NULL)
    print 5
    visit right of 5: node 9
      visit left of 9 (NULL)
      print 9
      visit right of 9 (NULL)
  print 12
  visit right of 12: node 18
    visit left of 18: node 15
```

```
        visit left of 15: node 13
          visit left of 13 (NULL)
          print 13
          visit right of 13 (NULL)
        print 15
        visit right of 15: node 17
          visit left of 17 (NULL)
          print 17
          visit right of 17 (NULL)
      print 18
      visit right of 18: node 19
        visit left of 19 (NULL)
        print 19
        visit right of 19 (NULL)
end
```

Similarly, the ***preorder tree walk*** displays the key of the root node of a subtree before it's left subtree and right subtree and the ***postorder tree walk*** displays the key of the root node after its left and right subtree. To easily remember the process, here is a summary of the keywords:

1. Inorder tree walk - *left*, *root*, *right*
2. Preorder tree walk - *root*, *left*, *right*
3. Postrder tree walk - *left*, *right*, *root*

Tasks

For this exercise, the following functions are to be implemented:

1. `void preorderWalk(BST* B)` - displays a list of the keys in the BST using preorder tree walk
2. `void inorderWalk(BST* B)` - displays a list of the keys in the BST using inorder tree walk
3. `void postorderWalk(BST* B)` - displays a list of the keys in the BST using postorder tree walk

Instructions

1. Add the following prototypes in `BST.h` (documentation for each function is also included below):

```
/*
** function: preorderWalk
** requirements:
    a non-null BST pointer
** results:
    displays a list of elements of the BST using `pre-order traversal`
*/
void preorderWalk(BST* B);

/*
** function: inorderWalk
** requirements:
    a non-null BST pointer
** results:
    displays a list of elements of the BST using `in-order traversal`
```

```
*/
void inorderWalk(BST* B);

/*
** function: postorderWalk
** requirements:
    a non-null BST pointer
** results:
    displays a list of elements of the BST using `post-order traversal`
*/
void postorderWalk(BST* B);
```

2. Implement the three traversal functions in `BST.c`
3. Create also test plans for these functions to check your implementation.

Learn to test your code and *as much as possible*, avoid submitting code with compile errors *i.e.* code that don't even run.

Notes on Files

To help you in your implementation, we include these two files:

- `BST.h` - a new version of `BST.h` where three new functions are included
- `main.c` - interpreter program now includes commands for these traversal algorithms; see description below.

Shell Program

A shell program is created to easily interact with the BST ADT. The available commands are described below:

- '+ `Xinserts the integer keyX`' in the BST.
- `? X` displays the location of the node with key `X`, if found.
- `p` reports the contents of the BST in tree mode.
- < lists the keys of the BST using preorder traversal.
- = lists the keys of the BST using inorder traversal.
- > lists the keys of the BST using postorder traversal.
- `Q` terminates the program.

Submission

Submit to our Lab Google Classroom a compressed (`.zip`) folder named `<CompleteLabSection><Surname>Ex04InLab.zip` (e.g.U1-1LDelaCruzEx04InLab.zip) It should contain the following:

1. `BST.h` and `BST.c`
2. `main.c`
3. `program.cs`
4. `Makefile`

Questions?

If you have any questions, approach your lab instructor.