Programming Assignment 6
Binary Search Tree

Due Date : Section 0 - Wednesday April 25th , 2018 - No Later than 2:15 pm
Section 1 - Wednesday April 25th , 2018 - No Later than 3:45 pm.
Section 2 - Wednesday April 25th , 2018 - No Later than 5:15 pm.

Modify the BST program that is discussed in the class by adding the following functions :

1. Write a function that builds a binary search tree by inserting the following nodes into the tree. (70 50 100 30 60 80 110 20 68 90 120 25 62). The function must display the binary search tree.
2. Write a function that traverse and displays the tree in pre-order form.
3. Write a function that traverse and displays the tree in post - order form.
4. Write a function that displays all the right sub roots of that tree.
5. Write a function that displays all the left sub roots of that tree.
6. Write a function that counts and displays the number of leafs in the tree.
7. Write a function that displays only the leafs of the tree.

Note :

- You have to use **only** dynamic arrays when implementing this program (items 1- 7 where applicable). Cannot use fixed size array , or linked lists.
- Cannot use library functions such as swap, sort , vector … etc. for this programming assignment.
- In the sample output , Replace My name (Husain Gholoom) with your first and last name.

Style Guidelines:

At the beginning of your program (and **before** the #include statement), include the following :

Header comments (file documentation block) should be at the top of each file and should contain: Author / s, Due Date, Assignment Number, Course number and section, Instructor, and a brief description of the purpose of the code in the file. For example :

```
//      Roster Number / s  :        xxxxxxxxx
//
//      Author / s : (Your name  here!!)
//      Due Date  :
//      Programming Assignment Number  6
//
//      Spring  2018  -  CS 3358  -  Your Section  Number
//
//      Instructor:   Husain Gholoom.
//
//       <Brief description of the  purpose of the program>
```

Variable names :

- Must be meaningful.
- The initial letter should be lowercase, following words should be capitalized, no other caps or punctuation (i.e. weightInPounds).
- Each variable must be declared on a separate line with a descriptive comment.

Named constants :

- Use for most numeric literals.
- All capitals with underscores (i.e. TX_STATE_SALES_TAX)
- Should occur at top of function, or global (only if necessary)

Line length of source code should be no longer than 80 characters (no wrapping of lines).

Indentation :

- Use 2-4 spaces (but be consistent throughout your program).
- Indent blocks, within blocks, etc.
- Use blank lines to separate sections.

Comments for variables :

All variable definitions should be commented as follows:

```
int  gender;    // integer value for the gender,
                        // 1 = Male , 2 = Female ,
```

Note : In order to get a full mark :

1. Your program **must compile** and run. We will test your program using latest version of codeblocks that is availed in Texas State University's Labs. Your program should run for any number of node.
2. Your program must be **documented according the style above . See the website for the sample programming style program.**
3. Must properly format the output as shown below.
4. Must **use functions (prototypes and definitions) .**
5. You must name your program as :

   - **LastName_FirstName_ PG6_BSTs.cpp**

   Where LastName is your Last Name and FirstName is your First Name. For example , the file name should look something like : Gholoom_Husain_PG6_BSTs.cpp (not .cbp)

6. Everyone must upload the electronic version of the program no later than the starting of class time on the due date. **DO NOT** send your assignment solution via email. **No late assignments will be accepted and a grade of zero will be assigned . Group members must upload identical copy of the assignment.**

   USE TRACS To upload your program .

7. You must **also** turn in hard copy of your source code no later than the starting of class time on the due date . Should the hard copy consist of more than one page , then , the hard copy must be **stapled**. If you are unable to turn in a printout during class, you can take the program to the computer science department and hand it to the front desk personal (Comal 211) before the deadline. Make sure that the front office stamps the program. Make sure that include the date and time. Finally ,make sure that they place the program in my mailbox. **Only one copy per group.**

   No late submission hard copy of the assignment will be accepted and a grade of zero will be assigned .

   DO NOT slide your program under my office door – It will **NOT** be accepted **and a grade of zero will be assigned .**

The following points will be deducted if :

- Compilation errors , Incorrect file format such as uploading .cbp instead of .cpp , missing electronic copy , missing the hardcopy , different copies of the assignment per group , using vector array or static array or linked lists , using function libraries such as swap / sort , more than one file such as using .h , .cpp file
  (- 10 points)

- Each logical error : **(- 1.5 points)**

- Other : **(1.75 points)** if **any** of the following takes a place :

  - Unable to read the source code due to unclear printing or printing in a landscape format.
  - Incorrect Output format.
  - Incorrect program file name.
  - Hard copy is not stapled.
  - Incorrect Style **such as but not limited to** Missing Header , footer , comments or program documentations , missing roster number , missing section number … etc

Use the following sample .cpp file to test your program

```
BinarySearchTree   tree(13);

cout << "Binary Search Tree By Husain Gholoom\n\n";
cout << "Inserting nodes.\n\n";
tree.insertNode(70);
tree.insertNode(50);
tree.insertNode(100);
tree.insertNode(30);
tree.insertNode(60);
tree.insertNode(80);
tree.insertNode(110);
tree.insertNode(20);
tree.insertNode(68);
tree.insertNode(90);
tree.insertNode(120);
tree.insertNode(25);
tree.insertNode(62);


cout << "Building  BST is completed.\n\n";

cout << "Values of the Binary Search tree.\n\n";



// Pre-Order Traversal .
cout << "Pre-Order Traversal of the BST :\n\n";
tree.preOrder();



// Post-Order Traversal .
cout << "Post-Order Traversal of the BST :\n\n";
tree.postOrder();


// All right sub root values.
cout << "Here are all right sub root values for the BST :\n\n";
tree.displayRSR();

// All left sub root values.
cout << "\n\nHere are all left sub root values for the BST :\n\n";
tree.displayLSR();
```

```
// Counting Number of Leafs .
cout << "\n\nNumber of Leafs  =     "  << tree.treeLeafsCount() ;

// Display the leaf values.
cout << "\n\nHere are the leaf values in the BST:\n\n";
tree.displayLeafValues();

cout << "\n\n\nApril 25 , 2018\n\nWritten by Husain Gholoom \n\n";
```

Sample Output Run

```
Binary Search Tree By Husain Gholoom
Inserting nodes.
Building  BST is completed.

Values of the Binary Search Tree  :

Pre-Order Traversal :

Post-Order Traversal :

Here are all right sub root values for the BST  :

Here are all left sub root values for the BST  :

Number of Leafs  :

Here are the leaf values in the BST :


April 25 ,  2018
Written by Husain Gholoom
```