Programming Exercise 05
Binary Search Tree

OBJECTIVE

To be able to implement the tree ADT and its operations.

INSTRUCTIONS

1. You are to work on this activity individually or by pair. There should only be one submission per person/pair.
2. You are to write a code for a program that follows the details in the PROGRAM SPECIFICATION section. There should only be one program for this PE.
3. You are to develop your program following the structured programming approach at the very least. No global variable declarations are allowed.
4. All your program files (source code files, input files, etc.) should be together with your main program file (the one that contains the main function). There should be no need to put certain files in certain folders just so your program will compile and run.
5. You only need to submit the source code file you have created. If you created more than one source code file (i.e. source code is broken down into several files) or there are other files (e.g. input files) involved, archive them in a single zip file. Name your source code file or zip file, whichever is applicable, using your surname followed by the PE # similar to this example: Rizal_05 (for individual) or Bonifacio_Rizal_05 (for pair/group).

PROGRAM SPECIFICATION

The tree program will have the following main menu:

Binary Search Tree
[1] Insert item
[2] Delete item
[3] Search item
[4] Find maximum
[5] Find minimum
[6] Find successor
[7] Find predecessor
[8] Display tree
[0] Exit

Enter choice:

Details of the menu items are as follows:

Insert item
Once selected, the program should ask the user for a number to be inserted. The program should then search for the position where the item will be inserted. If a node that contains the

same item already exists, simply add one to the count of the item; otherwise, create a node to contain the item.

Delete item
Once selected, the program should ask the user for a number to be deleted. The program should then search for the item in the tree.

If the item does not exist, simply inform the user. Else, do whichever is applicable: If the item count of the node containing the item is more than one, simply update the count; otherwise, perform deletion of the node.

Search item
Once selected, the program should ask the user for an item to be searched. Search the item in the tree and display a message appropriate to the result of the search. If the item exists, display also the count.

Find maximum
Once selected, the program should display the maximum item found in the tree.

Find minimum
Once selected, the program should display the minimum item found in the tree.

Find successor
Once selected, the program should ask the user for an item. The program should then search the item in the tree. If the item does not exist, simply inform the user; otherwise, the program should then search for the successor of the item. If the successor does not exist, simply inform the user; otherwise, display the successor.

Find predecessor
Once selected, the program should ask the user for an item. The program should then search the item in the tree. If the item does not exist, simply inform the user; otherwise, the program should then search for the predecessor of the item. If the predecessor does not exist, simply inform the user; otherwise, display the predecessor.

Display tree
Once selected, the program should display the content of the tree. Display should be in three ways: pre-order, in-order, and post-order.

Exit
Once selected, the program should terminate with a message informing the user that the order system has been terminated.

Important:
1. If the process is to be performed on an empty tree, simply inform the user that the tree is empty.
2. At the end of each process, the program should display a message appropriate to the result of the process.
3. After processing the selected main menu item, the program should loop back to the main menu. The program should end only when the user selects the main menu item ***Exit***.