Programming Assignment 8

CSCD300 Data Structure

Instructor: Dr. Bojian Xu
Eastern Washington University, Cheney, Washington

Due: 11:59pm, December 5, 2020 (Saturday)

Please follow these rules strictly:

1. Verbal discussions with classmates are encouraged, but each student must independently write his/her own work, without referring to anybody else's solution.
2. No one should give his/her code to anyone else.
3. The deadline is sharp. Late submissions will **NOT** be accepted (it is set on the Canvas system). Send in whatever you have by the deadline.
4. Every source code file must have the author's name on the top.
5. All source code should be commented reasonably well.
6. Sharing any content of this assignment and its keys in any way with anyone who is not in this class of this quarter is NOT permitted.

Notice: I know you can easily find BST implementation code on the Internet and in fact I encourage you to learn it from the Internet. However, if you do so, make sure you write your own code for every line of the submission to this assignment. If you simply copy and paste and even with some modifications (for example, change the variable names and code structure and style, and so on, which the grader and I will be able to tell) from the Internet, you will get a clean zero for this assignment.

Total: 100 points.

Implementing a full package of binary search tree

In this programming assignment, we want to implement a binary search tree class that supports a variety of operations. For simplicity, we make each tree node have only four attributes [1]:

```
class BST_Node{
   int key;           // the key saved in the node
   BST_Node left;     // the reference to the left child node if it exits; null, otherwise
   BST_Node right;    // the reference to the right child node if it exits; null, otherwise
   BST_Node parent;   // the reference to the parent node if it exits; null, otherwise
```

[1] Only pseudocode is provided in this handout.

```
   /* constructor */
   BST_Node{int k}{
      key = k;
      parent = left = right = null;
   }

   /* Supporting methods are here */
}
```

The following pseudocode gives the BST class prototype that we want to implement. You can use the algorithmic ideas that we have discussed in the lectures to implement all the class member functions.

```
class BST{
   BST_Node root;   //the root of the BST

   /* constructor */
   BST(){
      root = null;
   }

   /* If the given key k exists in the BST, return the reference
      to the node that contains k; otherwise, return null;
   */
   BST_Node search(int k){
      ...
   }

   /* If the given key k already exists in the BST, return null;
      otherwise, insert k into the BST and return the reference
      to the new node that contains k.
   */
   BST_Node insert(int k){
      ...
   }

   /* If the given key k already exists in the BST, delete the node
      that contains k from the BST and return the reference to the
      deleted node; otherwise, return null.
   */
   BST_Node delete(int k){
      ...
   }

   /* Inorder traversal and print all the nodes in the subtree rooted
      at "subtree_root".
   */
   void InOrder_Traversal(BST_Node subtree_root){
      ...
   }

   /* Preorder traversal and print all the nodes in the subtree rooted
      at "subtree_root".
   */
   void PreOrder_Traversal(BST_Node subtree_root){
      ...
   }
```

```
    /* Postorder traversal and print all the nodes in the subtree rooted
       at "subtree_root".
    */
    void PostOrder_Traversal(BST_Node subtree_root){
       ...
    }

    /* Level-order traversal and print all the nodes in the subtree rooted
       at "subtree_root".
    */
    void LevelOrder_Traversal(BST_Node subtree_root){
       ...
    }

   /* return the reference to the node that contains the smallest key
       in the subtree rooted on "subtree_root".
    */
    BST_Node Min(BST_Node subtree_root){
       ...
    }

    /* return the reference to the node that contains the largest key
       in the subtree rooted on "subtree_root".
    */
    BST_Node Max(BST_Node subtree_root){
       ...
    }

    /* Return the reference to the node whose key is the successor of
       the key in "node", if such successor exists; otherwise, return null.
    */
    BST_Node Successor(BST_Node node){

    }

    /* Return the reference to the node whose key is the predecessor of
       the key in "node", if such successor exists; otherwise, return null.
    */
    BST_Node Predecessor(BST_Node node){

    }
}
```

The specification of your program

1. Your program should be named as: `BST.java`
2. The input and output of your program.

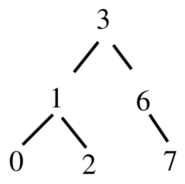
   The input is a pure ASCII text file, where each line is an integer number. It is possible that **duplicates** are existing in the input file. Your program will then read in this file and construct a BST for all the **distinct** numbers in the file by inserting them one by one. For example, if we supply to your program with a file named `data.txt` with the following content,

```
3
6
1
6
0
7
2
```

Then, the command line you should type would be:

```
$java BST data.txt
```

Your program will then process the input file and create the following BST in your main memory:



The output of your program is an interactive menu based user interface, which we will describe in more details next.

An interactive menu-based user interface.

After your program constructed the BST for the data from the input file, your program will then print the following menu items and options and wait for the user's choice.

```
Choose one of the following options.
====================================
1) Search for a key
2) Insert a new key
3) Delete an existing key
4) Inorder traversal of the BST
5) Preorder traversal of the BST
6) Postorder traversal of the BST
7) Level-order traversal of the BST
8) Find the smallest key
9) Find the largest key
a) Find the successor of a given key
b) Find the predecessor of a given key
x) quit

Your choice:
```

1. If the user chooses `1`, your program will prompt

```
Input the key:
```

The user will then type the key. If the given key exists, your program will print the following message and then goes back to print the menu items.

```
The given key exist.
```

If the given key does not exist, your program will print the following message and then goes back to print the menu items.

```
The given key does not exist.
```

2. If the user chooses `2`, your program will prompt

   ```
   Input the key:
   ```

   The user will then type the key. If the given key exists, your program will print the following message and then goes back to print the menu items.

   ```
   The given key already exists.
   ```

   If the given student id does not exist, your program will insert the given key and print the following message and then goes back to print the menu items.

   ```
   The given key has been inserted successfully.
   ```

3. If the user choose `3`, your program will prompt

   ```
   Input the key:
   ```

   The user will then type the key. If the given key exists, your program will print the following message and then goes back to print the menu items.

   ```
   The given key has been successfully deleted.
   ```

   If the given key does not exist, your program will print the following message and then goes back to print the menu items.

   ```
   No such a key.
   ```

4. If the user chooses `4`, your program will print all the keys in the tree using the inorder traversal and then goes back to print the menu items.
5. If the user chooses `5`, your program will print all the keys in the tree using the preorder traversal and then goes back to print the menu items.
6. If the user chooses `6`, your program will print all the keys in the tree using the postorder traversal and then goes back to print the menu items.

7. If the user chooses `7`, your program will print all the keys in the tree using the level-order traversal and then goes back to print the menu items.

8. If the user chooses `8`, your program will print the smallest key in the tree, if the tree is not empty; otherwise, your program will print the following message.

   ```
   The tree is empty.
   ```

   Your program will then go back to print the menu items.

9. If the user chooses `9`, your program will print the largest key in the tree, if the tree is not empty; otherwise, your program will print the following message.

   ```
   The tree is empty.
   ```

   Your program will then go back to print the menu items.

10. If the user chooses `a`, your program will prompt

    ```
    Input the key:
    ```

    The user will then type the key. (1) If the given key and its sucessor both exist, your program print the successor key's value and then goes back to print the menu items; (2) If the given key does not exist, your program will print the following message and then goes back to print the menu items.

    ```
    No such a key.
    ```

    (3) If the given key does exist, but the given key does not have a successor, your program will print the following message and then goes back to print the menu items.

    ```
    The given key exists but does not have a successor.
    ```

11. If the user chooses `b`, your program will prompt

    ```
    Input the key:
    ```

    The user will then type the key. (1) If the given key and its predecessor both exist, your program print the predecessor key's value and then goes back to print the menu items; (2) If the given key does not exist, your program will print the following message and then goes back to print the menu items.

    ```
    No such a key.
    ```

    (3) If the given key does exist, but the given key does not have a predecessor, your program will print the following message and then goes back to print the menu items.

    ```
    The given key exists but does not have a predecessor.
    ```

12. if the user choose `x`, your program will just quit.

Submission

- All your work files must be saved in one folder, named: **firstname_lastname_EWUID_cscd300_prog8**
  (1) We use the underline '_' not the dash '-'.
  (2) All letters are in the lower case including your name's initial letters.
  (3) If you have middle name(s), you don't have to put them into the submission's filename.
  (4) If your name contains the dash symbol '-', you can keep them.
- You only need submit java source code file(s).
- You then compress the above whole folder into a .zip file.
- Submit .zip file onto the Canvas system by the deadline.