Lab5 Assignment – Binary SearchTrees

Do these problems in order. While implementing/debugging, hard-code the program's input in your file. Once your code is working you can prompt the user for input.

Problem 1: BST operations.

Implement the Ordered Dictionary ADT operations using Binary Search Trees. Write code to implement the **`search, minimum, maximum, successor, predecessor, insert, delete`** operations. You can use the following TreeNode structure for nodes in the BST.

```
struct TreeNode {
   int value;
   struct TreeNode* left;
   struct TreeNode* right;
 }
```

To test your code, you can create a BST in the main (or in a separate function) and run the operations on that tree. Implement a menu-driven code program with difference options to execute the different functions you have implemented. You could have a separate **`buildTree`** function that will take as input a sequence of integers and insert them one by one into a BST by repeatedly calling the **`insert`** function.

To print a BST, you can define & use a method **`preorder-traverse`** that prints the pre-order traversal of the keys in the BST.

Problem 2: Parse Trees.

a) Write a program to build a parse tree from an fully parenthesised expression.
b) Write functions **`printPostfix`** and **`printPrefix`** that take as input an expression parse tree and prints the postfix and prefix representation of a parse tree.
c) Write a function **`evaluateTree`** to evaluate the value of a expression parse tree.

Problem 3.* Print the *level order traversal* of a binary tree. Given a binary tree, you should first print the root (level 0), then its children (level 1), then their children (level 2),..and so on. Within a given level, the nodes should be printed from left to right.