CS 3305: Data Structures
Spring 2021

Assignment 9 – Trees
100 points

Note: If you re-upload the files, you must re-upload **ALL** files as the system keeps the most recent uploaded submission only.
Note 2: Never hard-code test data in the test program. Always allow the user to enter the test data using menu option.

The goal of this assignment is to reinforce tree implementation in C++.

Part 1: (50 points)
For this assignment, you need the zip file *BST.zip*, included with this assignment. Extract the files in zip file BST and compile and run the test program testBST.cpp and understand the code and its output.

Next, develop a new test program, in file myTestBST.cpp, to build a binary search tree with the following words. Insert them in an order so that the resulting tree has as small a depth as possible. Use function *is_balanced()* to print out the tree depth. Display the final tree a tree-shape as demonstrated in the provide test program.

After	**first**	**only**	**two**
Also	**give**	**other**	**us**
any	**how**	**our**	**use**
back	**its**	**over**	**want**
because	**look**	**than**	**way**
come	**most**	**then**	**well**
day	**new**	**these**	**work**
even	**now**	**think**	

Next, remove the root of the tree; print out the tree depth; and re-display the tree in a tree-shape. Examine the output and make sure the new root is the correct value.

Part 2: (50 points)
A function of BST is to remove duplicate values from a data set. Develop a second test program, in file removeDouplicates.cpp. The program reads text input (words) from the keyboard and adds the words (not characters) to a BST, using space as the delimiter to extract tokens (words). Notice a word van be single character. The program then traverses the BST and prints out the words in order on the screen. Use the appropriate traversal function in the provided file *bintree.template*.

The following are sample runs showing output format. Do Not hard-code test data in the code. Do Not use output labels as input prompts. Test your code with the following sample inputs and format the output as shown below. Notice how output values are printed below the labels.

Sample test 1:

```
Original Text:
a B 2 n w C q K l 0 M a M l a B 2 n w 9

Processed Text:
0 2 9 B C K M a l n q w
```

Sample test 2:

```
Original Text:
this is a test for the test in the program that does no test

Processed Text:
a does for in is no program test that the this
```

Sample test 3:

```
Original Text from input file:
Following the test programs above, write a new test program called
RemoveDuplicates.java. The program reads text input from keyboard or a
text file and adds the words to a BST. The program then traverses the BST
and prints out the words in order on the screen (or to output text file).
Note that you may need to make some changes to BST.java.


Processed Text:
(or BST BST. BST.java. Following Note RemoveDuplicates.java. The a above,
adds and called changes file file). from in input keyboard make may need
new on or order out output prints program programs reads screen some test
text that the then to traverses words write you
```

This assignment is very specific and must be implemented as specified. Any deviation from these requirements will not be accepted and receives no points. No exceptions.

Do not forget to include author header in each submitted file as shown, **no header, no points!**

```
// Name:          <your name>
// Class:         CS 3305
// Term:          Spring 2021
// Instructor:    Dr. Haddad
// Assignment:    9
```

Submission:

Please submit all five files (bintree.h, bintree.template, bst.h, bst.template, myTestBST.cpp, and removeDouplicates.cpp) to the assignment submission folder in D2L by the due date posted in D2L. **Do NOT submit file testBST.cpp provided with the assignment. No late submissions are accepted**. Once again, please include **author header** block in each submitted file - **no headers, no points.**

Important Note: The code must be correctly running right and gives correct result in the required environment (CodeLite and GNU C++ compiler) before being uploaded.