UCR - CS 010C

Assignment 3

**Deliverables:** Create a single PDF file that contains your answers to the questions. Then create a zip file that contains this PDF file along with all your code source files. Submit this zip file on iLearn.

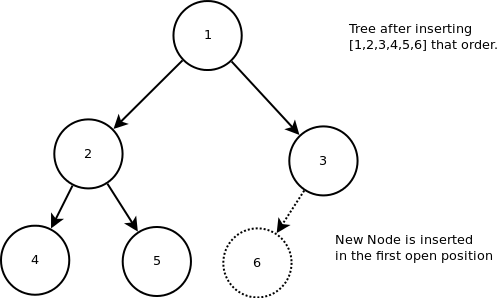
**Deadline:** 11/15/2020 11:59 pm.

Use provided C++ skeleton files (MyTree.cpp, MyTree.h and test.cpp) to insert your code.

1. Implement a binary tree class **MyTree** using pointers (define *struct* **BinaryNode** to store internal nodes), where each node has a pointer to each of its children and to its parent, and each node holds two variables, a string myString and an integer myInt.

Write functions in **MyTree** class to do the following:

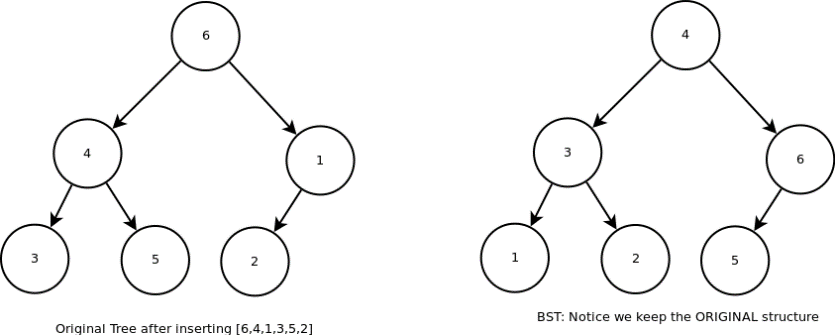
1. **Insert(string, int)**: Insert new node into first available position (to keep the tree almost complete). E.g.,



To get full points your Insert functions should be faster than , where is the number of nodes in the tree. Hint: you may use additional private variable(s) in MyTree class.

1. **Preorder()**: Output all strings in pre-order.
2. **FindMax()**: Return a pointer to the node with maximum myInt. (If multiple nodes have the same maximum myInt, return any node.)
3. **MakeBST()**: Convert the binary tree into a binary search tree (BST) with respect to myInt. That is, move around node values (myString and myInt) to satisfy the BST property. Do not change the structure of the tree (i.e. the pointers) but only swap myString and myInt values. (Hint: if you need to sort an array, you can use [std::sort](https://en.cppreference.com/w/cpp/algorithm/sort) method; use [std::swap](https://en.cppreference.com/w/cpp/algorithm/swap) to exchange the values of 2 variables)

e.g.,



1. What is the big-Oh complexity of your functions above? Also, what is the space complexity of your functions? Are they all in-place? If not, how much extra space do they need
   1. O(logn)
   2. O(logn)
   3. O(n)
   4. O(n)
   5. O(n)
   6. O(n)
   7. O(nlogn)
   8. O(n)
2. Test and measure the performance of your functions.

Create sequences of 100, 1000, 10000, 100000 random (string, int) pairs and insert them into MyTree (using Insert() function). Measure the times (in **nanoseconds** or **microseconds**) to

1. build the tree,
2. execute Preorder(),
3. execute FindMax(),
4. execute MakeBST().

Report the times for each tree size in a table.

Hint: To generate *N* random unique numbers, you can first create an array or a vector with *N* unique numbers (e.g., 1 to *N*), then use [std::shuffle](http://www.cplusplus.com/reference/algorithm/shuffle/) to rearrange them in a random order.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |