**Binary Search Trees**

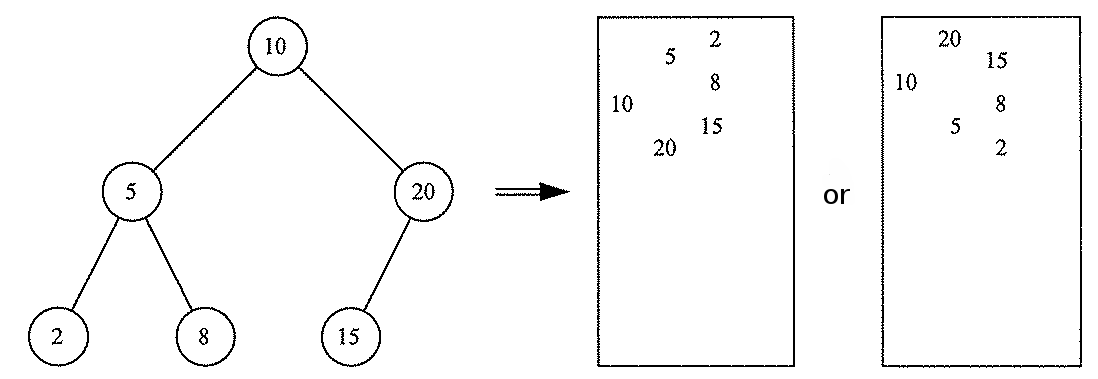
**Due: Thursday 11/19, 11:55 pm**

**Goal**

1. Understanding Binary Search Trees.

**Requirements:**

At the file [BST.cpp](http://cs.txstate.edu/~v_m137/cs3358_fall2015/assign5/BST.cpp) you will find part of a program that handles Binary Search Trees. At the program, the function bst\_insert(), which is used to insert items to the BST has been implemented. You are asked to write the following methods:

1. Write the method bst\_search() (with appropriate parameters), which can be used to check if an integer key exists in the tree or not. The method should return a reference (pointer) to the node that contains the key or NULL if the key does not exist in the tree.
2. 
3. if(node != NULL) //If the node is not NULL...
4. {
5. //If we have found the node...
6. if(item == node->data())
7. return node; //Return it.
8. //If the node's data is greater than the search item...
9. if(item < node->data())
10. //Search the left node.
11. return p\_search(item, node->left());
12. //If the node's data is less than the search item...
13. else
14. //Search the right node.
15. return p\_search(item, node->right());
16. }
17. else
18. //If the node is NULL, return NULL.
19. return NULL;
20. }
21. // Base Cases: root is null or key is present at root
22. if (root == NULL || root->key == key)
23. return root;
25. // Key is greater than root's key
26. if (root->key < key)
27. return search(root->right, key);
29. // Key is smaller than root's key
30. return search(root->left, key);
31. }
32. Write the method bst\_delete() (with appropriate parameters), which can be use to delete a key from the tree, if that key exists in the tree. If the key does not exist in the tree, it should print a message and return. The method returns a reference (pointer) to the root of the tree in either case.
33. Write the method print\_binary\_tree() (with appropriate parameters), which prints they keys of the tree as shown bellow. Each line should contain one integer. The root of the tree should be at the left and the leaves at the right. Hint: use recursion.  
    
34. Write the method free\_binary\_tree() (with appropriate parameters), which frees the memory occupied by the BST by deleting all the nodes of the tree, starting from the leaves and going all they way up to the root.

Run and test your program as follows:

* print the contents of the tree
* insert the integers 20, 30 and 23 in that order
* print the contents of the tree
* insert the integers 8, 1 and 26 in that order
* print the contents of the tree
* insert the integers 15, 40, 4, 28, 2 and 25 in that order
* print the contents of the tree
* search for the keys 15, 25, 35
* delete the keys 15, 1, 32 and 20 in that order
* print the contents of the tree
* free the nodes of the BST
* print the contents of the tree
* terminate the program

Save the results of the the above steps in a document and submit it together with your source code.

**Notes:**

* You must do this program by yourself.
* Hand in a zipped file named **prog05\_xxxxxx.zip** where xxxxxx is your TXstate id number.
  + Include the **BST.cpp** file.
  + Include the **Output.txt** file.
* Be sure to follow the [documentation standards](http://cs.txstate.edu/~v_m137/cs3358_fall2015/codingstyle.html) for the course.

**Turn in**: No hard copy source file turn in.

**Submit**: using [TRACS](https://tracs.txstate.edu/)