Data Structure and Algorithm Analysis---COP3530

Program –Module 11

Total Points: 25

After completing this assignment, you will be able to implement a binary search tree (BST).

In this assignment you will implement a program to manage a binary search tree (BST). You will need to make several modifications to the BST class the I provided in the file "bst.cpp". Name the driver for this assignment "bst_driver.cpp". Consider the following changes.

1. Change the declaration of treenode to the following:

```
class treenode //node in a bst
{
public:
     string county_name;
     double population_size;
     treenode* lchild, * rchild; //left and right children pointers
};
```

2. Make the following changes to the BST class declaration in the file in bst.cpp. (Remember, to put the declaration in a separate file (bst.h).

    a) change the name from "BST" to "bst";
    b) change default constructor from "BST" to "bst";
    c) change name of "insert_aux" to "insert";add appropriate formal parameters;
    d) change name "del" to "del_name";
    e) change name of "del_aux" to "del_name" ; add appropriate formal parameters;
    f) change name of "search_tree_aux" to "search_tree"; add appropriate formal parameters;
    g) change name of "print_tree_aux" to "print_tree"; add appropriate formal parameters;
    h) add the prototypes for "sorted_info"; see class declarations below;
    i) add the prototype for "empty_tree"; see declaration below;
    j) add the prototypes for "sorted_info"; see declaraton below;

After you make the changes to the BST class, the declaration in "bst.cpp" should look like the following:

```
class bst
{
public:
	bst(); //store the data file ("county_data.txt") into initial bst.
	~bst();//de-allocates dynamic memory allocate for tree bool empty(); // checks to see if the
	                                                                    //tree is empty.
	bool empty(); //return true if bst is null; otherwise false is retured.
	void insert(const string& item, const double& population); //auxiliary function.
	void insert(treenode*&, const string& item, const double& population); //inserts an item into
	       //the bst; this function is invoked, in main, by the auxiliary function described above.
	void del_name(string item); //auxiliary function.
	void del_name(treenode*& loc_ptr, string item); //function deletes an item from the bst; this
	                       //function is invoked, in main, by the auxiliary function described above.
	 treenode* search_tree(string item); //auxiliary fucntion
	treenode* search_tree(treenode*, string item); //returns a pointer to the node in bst with a
	            //county name county name that matches item; this function is invoked, in main, by
	            //the auxiliary function describes above.
	treenode* inorder_succ(treenode*); //returns a pointer to the inorder successor, based on county
	                                   //name.
	void county_ranges(const string& min_name, const string& max_name); //auxiliary function.
	void county_ranges(treenode*&, const string& min_name, const string& max_name); //prints all
	                      //the county names to the screen that lie between min_name and max_name,
	                      //inclusive.
	void print_tree();//auxiliary function.
	void print_tree(treenode*&);//prints each county record to the screen sorted by county name.
	void sorted_info();//auxiliary function.
	void sorted_info(ostream&, treenode*&);//prints each county record to a file called
	                                       //"county_info.txt" sorted by county name".
	void empty_tree();//de-allocate all nodes that were allocated to the bst
private:
	treenode* root;
};
```

You may add more functions to the class if necessary.

Note that the auxiliary functions are invoked inside main. The main purpose of the functions is to allow access to the bst by invoking a function with the same name that has a pointer to the root of the bst. Remember, the root is in the state of the class, and cannot be accessed outside the class.

You should submit your files (bst.h, bst.cpp, and bst_driver.cpp) to Canvas before the due date and time. Start early.