CPSC 2430 Data Structure

Winter Quarter 2024

Lab 3 - BST Basic Operations **Due: 10:00 pm, Tuesday, Jan 23**

Lab 3 focuses on BST basic operations using a linked structure and recursion.

The veterinary team at the Animal Shelter would like to be able to search for pets by "age" so that they can give them the proper vaccines. You will be using the following definitions to implement basic operations recursively for a binary search tree class containing pointers to Pet objects.

```
/*
** Binary Search Tree Basic Operations
** Lab 3, CPSC 2430
*/

class ShelterBST {
  private:
    struct Pet {
      string name;
      int age;
      // add your constructor here
      // ...
    };
    struct TreeNode {
        Pet* pet;          // you must use a Pet pointer
        TreeNode* left;
        TreeNode* right;
    };

    TreeNode* root;

    // PRIVATE RECURSIVE FUNCTIONS
    // For this lab, TreeNodes will be ordered by Pet age.
    // No duplicate values (Pet ages) will be entered for this lab.
    TreeNode* insert(TreeNode * root, Pet* pet) {
        // add your code here
        // ...
    }
```

```
  // Returns pointer to TreeNode that matches the given age
  // Nullptr if no match is found
  TreeNode* search(TreeNode *root, int age) {
      // add your code here
      // ...
  }

  // The three traversals below will neatly print to screen
  // the Pets' names and ages in the respective order
  void inorder(TreeNode * root) {
      // add your code here
      // ...
  }

  void preorder(TreeNode * root) {
      // add your code here
      // ...
  }

  void postorder(TreeNode * root) {
       // add your code here
       // ...
  }

public:
  ShelterBST() {
     root = nullptr;
  }
  void insertPet(string name, int age){
    root = insert(root, new Pet(name, age));
  }
  void searchPet(int age){
    TreeNode* result = search(root, age);
    // display name of pet found or
    // message if not found
    // ...
  }
  void inorderDisplay(){
    inorder(root);
  }
  void preorderDisplay(){
     preorder(root);
  }
  void postorderDisplay(){
     postorder(root);
  }
```

```
};

// In your main function, test your implementation
// 1. Insert 10 pets into your BST (with different ages)
// 2. Display the three traversals of the resulting BST
// 3. Conduct one successful search and one unsuccessful search
int main() {
  ShelterBST tree;

  // insert 10 pets, for example: (this syntax may be different
  // depending on how you design your constructors)
  tree.insertNode (“Zelda”, 5);
  tree.insertNode (“Link”, 7);
  // etc ...

  // inorder display
  tree.inorderDisplay();

  // preorder display
  tree.preorderDisplay();

  // postorder display
  tree.postorderDisplay();

  // successful search
  tree.searchPet(7);

  // unsuccessful search
  tree.searchPet(100); // assuming no Pet in your tree is aged 100

  return 0;
}
```

Lab 3 Submission

You need to submit the following file:

- lab3.cpp

Your lab3.cpp must include all the recursive functions mentioned above, with the exact same function prototypes (if you feel strongly about changing them, do reach out before the submission deadline). Before submission, you should ensure your program has been compiled and tested (extensively). Your assignment receives zero if your code cannot be compiled and executed.

You can submit your program multiple times before the deadline. The last submission will be used for grading.

To submit your assignment, run the script below from the directory containing your lab3.cpp file (assuming your files are on cs1.seattleu.edu):

```
/home/fac/mjilani/submit/24wq2430/lab3_submit
```

Lab 3 Grading Breakdown

Breakdown	Points	Note
insert	20	Your BST can be constructed successfully when adding adding nodes to it.
search	20	Find and print out the name of pet of given age (10pt), or a message if not found (10pt).
inorder traverse	20	Print out the inorder traversal of your BST
preorder traverse	20	Print out the preorder traversal of your BST
postorder traverse	20	Print out the postorder traversal of your BST
deductions (possible negative score)	0	Points will be deducted if you don't use pet pointers, are missing any of the required tests in main, use different function prototypes, etc. Points will also be deducted for lack of functional decomposition, use of global variables (not constants), inconsistent indentation, lack of comments, messy output, etc.

***For this lab, you do not need to de-allocate the Pets or TreeNodes you will create. We will take care of that in Assignment 3.**