**CS3100/5100: Data Structures and Algorithms**

**Programming Assignment #4**

1. Project Description

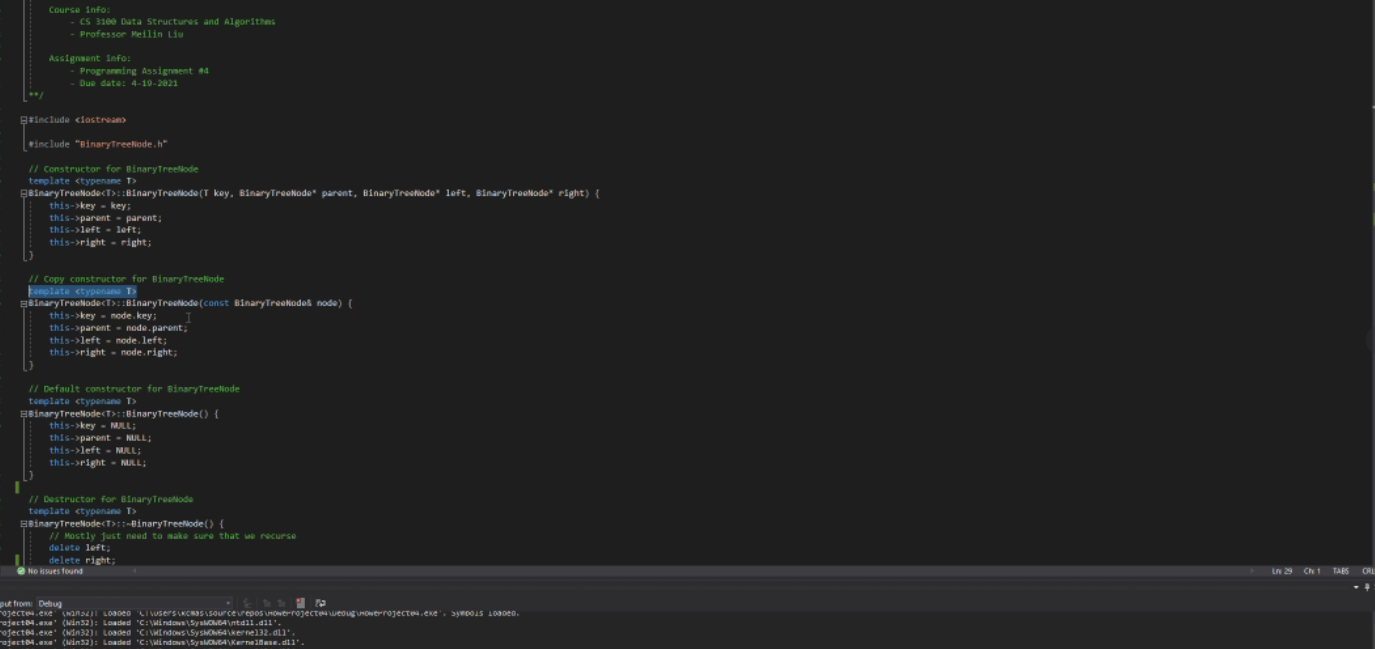
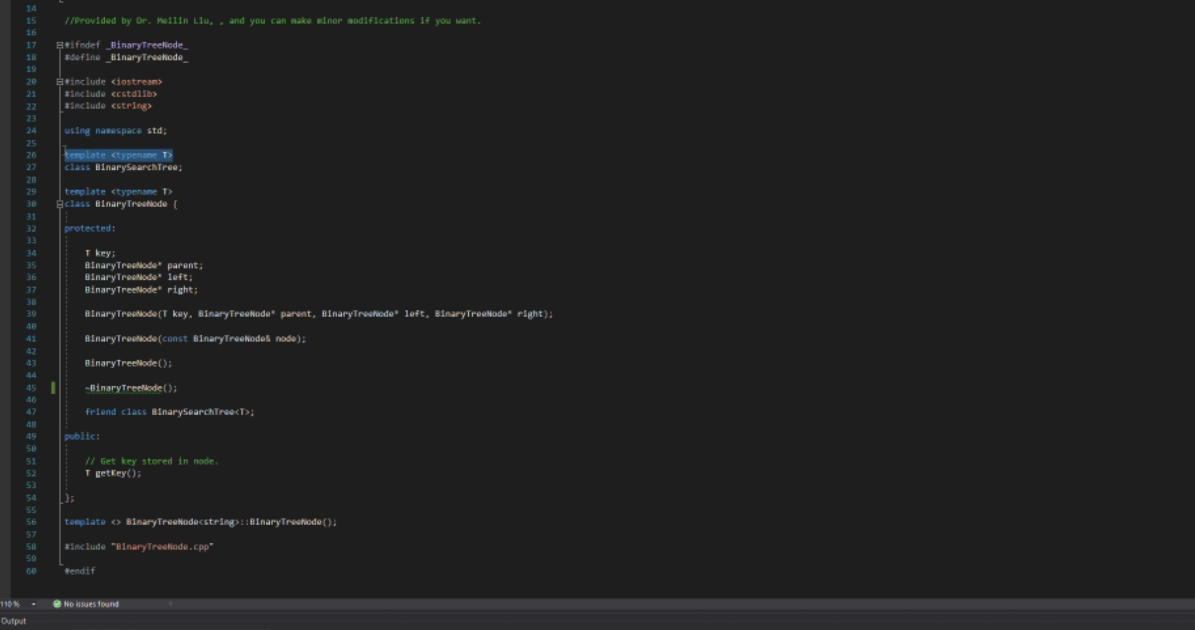
For this assignment, you will write a simple database for storing and retrieving last name records (string) using a Binary Search Tree. You will implement a Binary Search Tree in which each Binary Tree Node stores a last name. The Binary Search Tree is implemented in BinaryTreeNode.h, BinarySearchTree.h, and BinarySearchTree.cpp. The input database file that contains many lines of Last Names is provided. Duplicate last names are allowed in the database file, and will be inserted into the right subtree of the binary search tree.

In the beginning of your main() function, you should open the input database file, create a Binary Search Tree. Then you should read a specified number (20 – 50) of last name records from the input database file, and insert these last name records into the Binary Search Tree one by one using the Binary Search Tree’s insertion method (The last name will be used as the key for Binary Search Tree). Then you can print the employee records to the screen, using inorder tree traversal, and simultaneously save the employee records in the binary search tree to disk using inorder tree traversal.

Then you can test the following operations in the main() function:

• Insert a new last name: prompt the user for a last name, insert it into the Binary Search Tree. Please print out the last names stored in the binary search tree before the insertion operation, and after the insertion operation using inorder tree traversal.

• Delete a Record: Ask the user for a last name and delete it from the Binary Search Tree. Please print out the last names stored in the binary search tree before the delete operation, and after the delete operation using inorder tree traveral.

• Search: Print on the screen if the last name given via the keyboard is found or not. 

• Save the last name records in the binary search tree to disk using inorder tree traversal.

2. Requirements

The main.cpp file includes code that will test the implementation of your member functions of your Binary Search Tree ADT.

1. In BinarySearchTree.cpp, you must implement all the functions in BinarySearchTree.h.

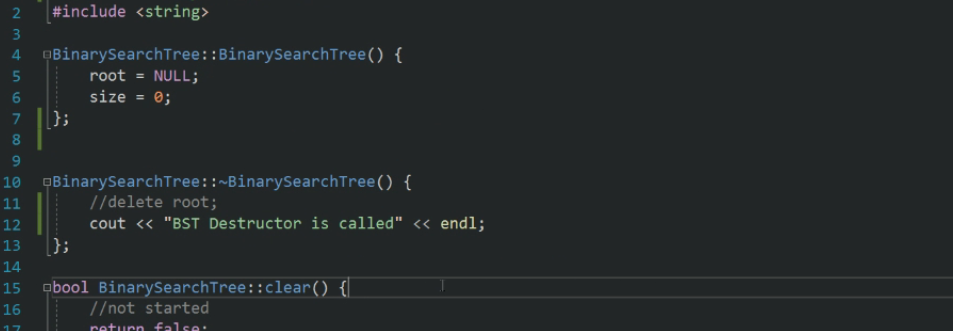
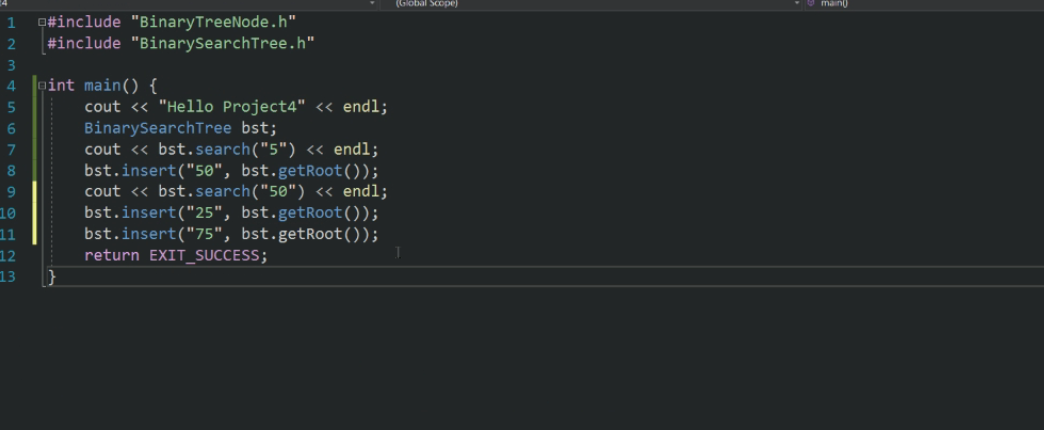
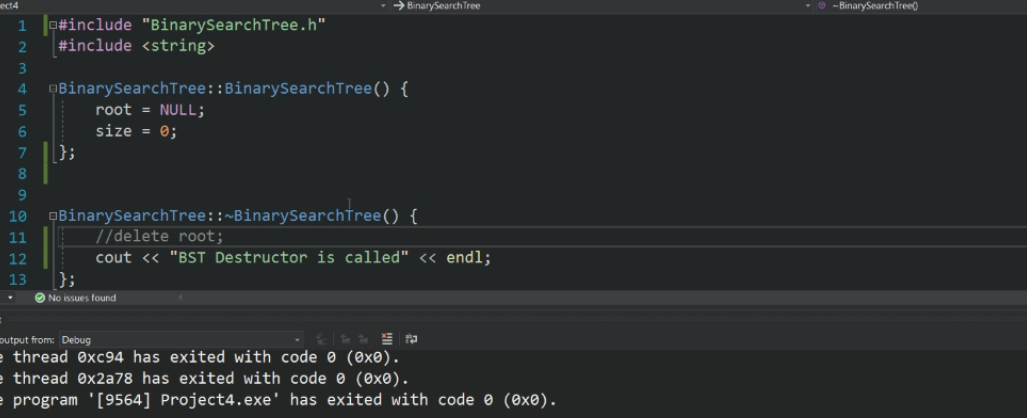
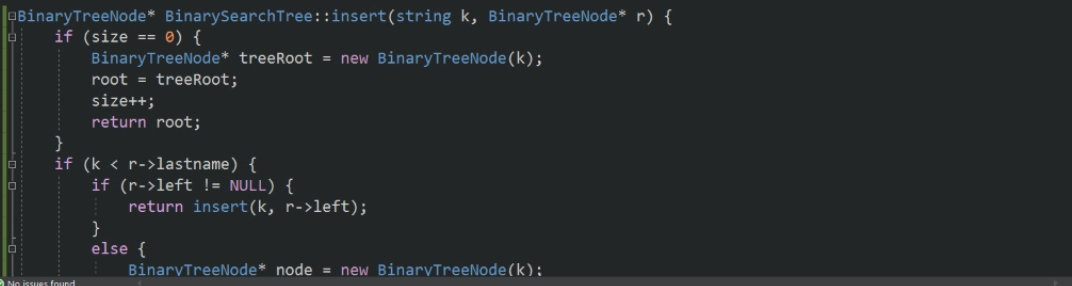
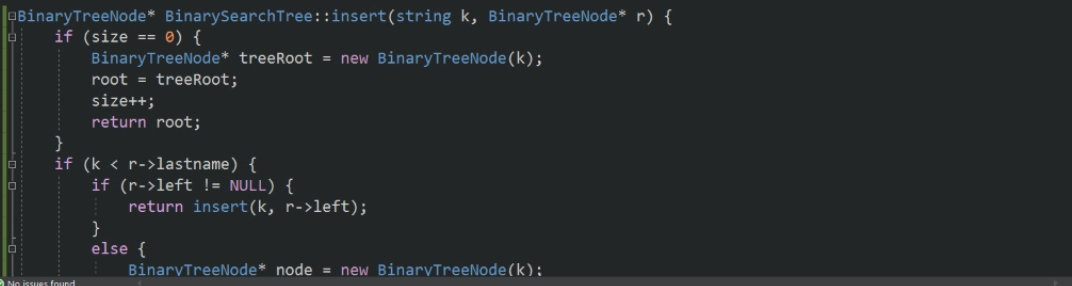
1. Your code should follow the Code Standards posted on Pilot. Your code will be graded according to its correctness, efficiency, organization, and readability.
2. Make sure that each file includes your name and email address in the header comments.

# Bonus Task

If your binary search tree class is implemented as a template class, that can not only support last name record, but also can support integers, doubles, strings, or other object types (i.e., the Binary tree node stores an integer, a double, a string, or an object of other object type), then you can get bonus points (20 points). You should test the template classes in your main.cpp file.

# Requirements

1. The GTA will test your source code using visual studio. It is suggested that you test your source code using visual studio.
2. You must submit an ELECTRONIC COPY of your source program through Pilot before the due date. If for some reason Pilot is unavailable, submit your source code to the instructor Meilin Liu, and the GTA.
3. Submit a zip file that includes all your source codes ( BinaryTreeNode.h, BinarySearchTree.h, BinarySearchTree.cpp, and lab3\_lastname.cpp), a README for your projects. You are required to submit a README file to tell the GTA what they need to know when testing your program. If your program cannot be compiled correctly, please report it in the README file. You can also state clearly what functionality of your project submission is not working correctly.
4. All the submitted project files should have: Course Number / Course Title, Your Name, Instructor’s Name, Date, and the Project Name. If you did not include these required contents in your submitted files, then 5 points will be deducted.
5. The GTA will test your programs under visual studio. It is YOUR responsibility to make your programs workable and runnable by others under school’s UNIX environment.
6. The programming assignment is individual. You must do the project by yourself. If you allow others to copy your programs or answers, you will get the same punishment as those who copy yours.
7. Be honest. If your program does not work, SAY SO in your README. The instructor will run your programs with ANY DIFFERENT input files. If your program does not work, but you “claim” yours work, you will have severe penalty!



//Base case if root is null or at desired value

if (root == NULL)

{

cout << "Name Not found";

return;

}

//If the key is found

if (root->lastname == lastname)

{

if (root->parent == nullptr)

{

cout << "The node with name " << lastname << " is root node";

}

else if (lastname < root->lastname)

{

cout << "The given name is the left node of the node with lastname" << root->lastname;

}

else

{

cout << "The given name is the right node of the node with lastname" << root->lastname;

}

return;

// if the given key is less than the root node, Use recursion along the left subtree.

if (lastname < root->lastname) {

return search(lastname, root->left);

}

// otherwise, recur for the right subtree

return search(lastname, root->right);

}