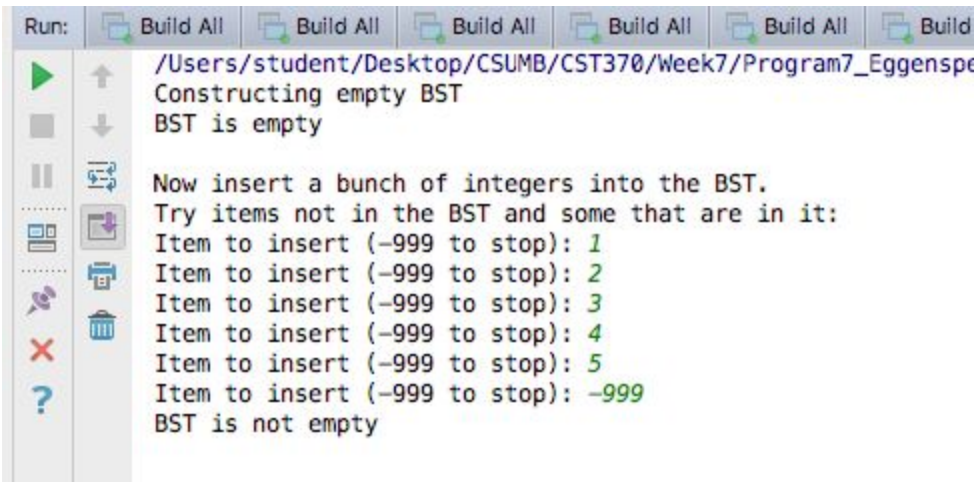
Mariya Eggensperger
CST 370, Spring 2017
Dr. Feiling Jia
Design/Analysis
of Algorithms

CST 370 : Programming Assignment (Binary Search Tree)

1. Download the source programs for Binary Search Tree (BST.cpp, BST.h, and Sample_BST_tester.cpp) from iLearn.

Figure 1 The initial program setup allows the user to input several nodes into the program and terminate the program upon completion.



(a) Change the current search() function to a recursive version.

As is seen in **Figure 2**, the function search() is recursive by virtue of its calling upon itself. No while-loop is thus present as was prior to recursive transformation.

Figure 2 Search() is recursive

```
bool BST::search(const int & item) const
{
    return searchRecursive(myRoot, item);
}
bool BST::searchRecursive(BinNode *locptr, const int &item) const {
    if (locptr == NULL)
        return false;
    if (item < locptr->data)        // descend left
        return searchRecursive(locptr->left, item);
    else if (locptr->data < item)   // descend right
        return searchRecursive(locptr->right, item);
    else                            // item found
        return true;
}
```

(b) Add a new member function called inOrder() that implements the inorder traversal algorithm of a binary search tree. Your function should display each node data on the screen.

void BST::inOrder()

In **Figure 3**, the inorder() output gives nodes in non-decreasing order, which is the denotation for an inorder traversal. The program, traverses the left subtree, visitis the root node, then traverses the right subtree.

Figure 3 inorder traversal outcome

```
Test the search() method by
inserting a number to search for.
Try items not in the BST and some that are in it:
Item to find (-999 to stop): 1
Found
Item to find (-999 to stop): 2
Found
Item to find (-999 to stop): -4
Not found
Item to find (-999 to stop): -999

--------------------TESTING--------------------

Testing inOrder(): 1 2 3 4 5
```

(c) Add a new member function called preOrder() that implements the preorder traversal algorithm of a binary search tree. Your function should display each node data on the screen.

void BST::preOrder()

In **Figure 4**, the preOrder() outcome creates a copy of the tree. Preorder traversal can also used to get prefix expression on of an expression tree. The algorithm visits the root node, traverses through the left subtree, then traverses through the right subtree.

Figure 4 preOrder() copy

```
Test the search() method by
inserting a number to search for.
Try items not in the BST and some that are in it:
Item to find (-999 to stop): 1
Found
Item to find (-999 to stop): 2
Found
Item to find (-999 to stop): -4
Not found
Item to find (-999 to stop): -999

---------------------TESTING---------------------

Testing inOrder(): 1 2 3 4 5
Testing preOrder(): 1 2 3 4 5
```

(d) Add a new member function called nodeCount() to count the number of nodes in a binary search tree. In this function, you should use a recursive function. You can't just use a variable such as "mySize".

int BST::nodeCount()

In **Figure 5**, the user has already input nodes into the program and receives a screen printout of the number of nodes present in the binary tree.

FIgure 5 Node count counts the node quantity present in the tree.

```
Test the search() method by
inserting a number to search for.
Try items not in the BST and some that are in it:
Item to find (-999 to stop): 1
Found
Item to find (-999 to stop): 2
Found
Item to find (-999 to stop): -4
Not found
Item to find (-999 to stop): -999

--------------------TESTING--------------------

Testing inOrder(): 1 2 3 4 5
Testing preOrder(): 1 2 3 4 5
Testing nodeCount = 5

Process finished with exit code 0
```

(e) UpdatetheSample_BST_tester.cpptoshowtheexecutionofthefunctions.

Driver updated to account for new member functions.

```
Run: Build All | Build All | Build All | Build All | Build All | Build All | Build All | Build All
/Users/student/Desktop/CSUMB/CST370/Week7/Program7_Eggensperger_Mariya/cmake-build-deb
Constructing empty BST
BST is empty

Now insert a bunch of integers into the BST.
Try items not in the BST and some that are in it:
Item to insert (-999 to stop): 1
Item to insert (-999 to stop): 2
Item to insert (-999 to stop): 3
Item to insert (-999 to stop): 4
Item to insert (-999 to stop): 5
Item to insert (-999 to stop): -999
BST is not empty


Test the search() method by
inserting a number to search for.
Try items not in the BST and some that are in it:
Item to find (-999 to stop): 1
Found
Item to find (-999 to stop): 2
Found
Item to find (-999 to stop): -4
Not found
Item to find (-999 to stop): -999
```

```
--------------------TESTING--------------------

Testing inOrder(): 1 2 3 4 5
Testing preOrder(): 1 2 3 4 5
Testing nodeCount = 5

Process finished with exit code 0
```