

## P2: BST

10/31/2023

95.5/100 Points

Offline Score:  
**95.5/100** Add CommentAnonymous Grading: **no**

## ▼ Details

Week 5: Algorithm Analysis (<https://seattleu.instructure.com/courses/1610311/pages/week-5-synopsis>)

ICE8: Big-O (<https://seattleu.instructure.com/courses/1610311/assignments/7157057>) >>> P2: BST P2x: BST (EC) (<https://seattleu.instructure.com/courses/1610311/assignments/7157075>)



## P2: BST

**All Projects (P)****Instructions:**

Implement a BST template class and write a client program to test it. The implementations of member functions must follow the declarations provided below.

**Class specification ( `BST.h` ):**

The following public functions must be defined in the BST template class.

- Constructor - Creates an empty BST of given element types.
- Destructor - Removes all elements from the BST.
- Copy constructor - Create a copy of the BST passed in as a parameter.
- Overloaded assignment operator - Assigns the BST passed in as a parameter to an existing BST.
- `insert` - Inserts the element passed in as a parameter into the BST. Assume no duplicate values. Must use recursion.
- `contains` - Searches the BST to determine if a given value is present. Returns true if so, else false. Must use recursion.
- `remove` - Removes the specified value from the BST.
- `empty` - Determines if BST is empty. Returns true if so, else false.
- `size` - Returns the count of nodes in the BST.
- `getLeafCount` - Returns the count of nodes in the BST that do not have any children.
- `getHeight` - Returns height of the BST. The height of the tree is the node with the greatest depth.

<https://seattleu.instructure.com/courses/1610311/modules/items/17916997><https://seattleu.instructure.com/courses/1610311/assignments/7157074>

- `getInOrderTraversal` - Returns a string of elements in the order specified by the in-order traversal of the BST. Must use recursion.
- `getPreOrderTraversal` - Returns a string of elements in the order specified by the pre-order traversal of the BST. Must use recursion.
- `getPostOrderTraversal` - Returns a string of elements in the order specified by the post-order traversal of the BST. Must use recursion.
- `getAncestors` - Returns a string of elements that are the ancestor(s) of the given node. The most immediate ancestor will be the first in the list. If the value is not present in the BST, returns an empty string.

### Notes / Hints:

- The implementations of member functions must match the public declarations provided in the class specification. Do not change the arguments, return values, or function names defined above.
- Do not add any additional `public` functions or fields to the class. You may create additional `private` helper functions or fields. Remember that helper functions may have the same name as the public function, but will need to have different arguments.
- You must use recursion to perform `contains`, `insert`, and the traversals. Remove can use an iterative or recursive solution.
- You will need to compare nodes that hold primitive data and strings. Be sure that you are able to compare these nodes in order for the `insert` function to work properly.
- You are not allowed to keep track of the number of elements in a private variable for `size`; you will need to traverse the entire BST to count the elements.
- You are not allowed to use any printing/reading from console in the `BST` class. (i.e. `cout`, `cin`, `printf`, etc.)
- You may want to `#include <sstream>` and use a stringstream object to capture the string traversals.
- You may use the integer BST class and code given in [Week3.zip](https://seattleu.instructure.com/courses/1610311/files/69761838/download?wrap=1) (<https://seattleu.instructure.com/courses/1610311/files/69761838/download?wrap=1>). [↓](https://seattleu.instructure.com/courses/1610311/files/69761838/download?download_frd=1) ([https://seattleu.instructure.com/courses/1610311/files/69761838/download?download\\_frd=1](https://seattleu.instructure.com/courses/1610311/files/69761838/download?download_frd=1)) as starter code.

### Test driver (`p2.cpp`):

Implement the driver program to thoroughly test the functionality of the BST.

- Ask the user for the file names to read.
- Reads the integers or strings from the file, and `insert` them into the BST in the order they appear
  - Note that the integer file being read should contain an integer on each line. The integers may be positive, negative, or zero.
  - Note that the string file being read should contain a string on each line. Spaces can be



- Thoroughly test the functionality of the remaining methods, for example:
  - Verify `contains` functionality by checking that some elements are present, but not others. See "Test files" section for info on what can be assumed to be included and not.
  - Verify `remove` functionality by removing some elements and that BST is modified as expected. Remove the elements that are noted in the "Test files" section.
  - Verify that `insert` functionality still works as expected after remove modified the BST. Insert the elements that are noted in the "Test files" section.
  - Verify that `getLevel` returns expected values in a known BST.
  - Verify that `getAncestors` returns expected values in a known BST.
  - As you add and remove elements, verify that `size`, `empty`, `getHeight`, and `getLeafCount` return expected values.

## Test files:

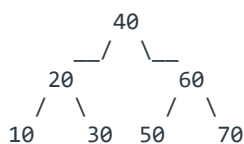
The test file will use a simple balanced BST to demonstrate, but your program will be tested with a file containing as many as 30 random integers or strings. Make sure to test thoroughly using your own input files.

### Contents of sample integer file

(`integers.dat`):

```
40
20
10
30
60
50
70
```

Inserting these values will result in the following BST:

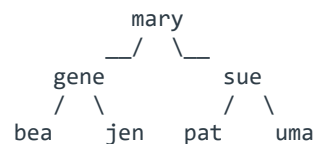


**Note:** Assume that 20, 40, 10, and 70 are always in the input file and that 99, -2, 59, and 43 are not in the input file. With this assumption, you can hard-code a list to test `contains`, `remove`, and `insert` (again).

### Contents of sample string file (`strings.dat`):

```
mary
gene
bea
jen
sue
pat
uma
```

Inserting these values will result in the following BST:



**Note:** Assume that "gene", "mary", "bea", and "uma" are always in the input file and that "yan", "amy", "ron", "opal" are not in the input file. With this assumption, you can hard-code a list to test `contains`, `remove`, and `insert` (again).

## Sample output (partial):

```
[username@cs1 p2]$ g++ -Wall -Werror -pedantic -std=c++14 p2.cpp -o p2
[username@cs1 p2]$ ./p2
```

Welcome (add more info about the program here)!

```
*****
* INTEGER BINARY SEARCH TREE *
```



(<https://seattleu.instructure.com/courses/1610311/modules/items/17916997>)

(<https://seattleu.instructure.com/courses/1610311/modules/items/17916997>)

```
BST height: -1
BST empty? true
```

Enter integer file: /home/fac/mthayer/submit/23fq5005/files/p2-files/integers.dat

```
** TEST INSERT **
```

```
Inserting in this order: 40 20 10 30 60 50 70
```

```
# of nodes: 7
# of leaves: 4
tree height: 2
tree empty? false
```

```
** TEST TRAVERSALS **
```

```
pre-order: 40 20 10 30 60 50 70
in-order: 10 20 30 40 50 60 70
post-order: 10 30 20 50 70 60 40
```

```
** TEST LEVEL & ANCESTORS **
```

```
level(40): 0, ancestors(40):
level(20): 1, ancestors(20): 40
level(10): 2, ancestors(10): 20 40
level(30): 2, ancestors(30): 20 40
level(60): 1, ancestors(60): 40
level(50): 2, ancestors(50): 60 40
level(70): 2, ancestors(70): 60 40
```

```
** TEST CONTAINS **
```

```
contains(20): true
contains(40): true
contains(10): true
contains(70): true
contains(99): false
contains(-2): false
contains(59): false
contains(43): false
```

```
** TEST REMOVE **
```

```
Removing in this order: 20 40 10 70 99 -2 59 43
```

```
# of nodes: 3
# of leaves: 2
tree height: 1
tree empty? false
pre-order: 50 30 60
in-order: 30 50 60
post-order: 30 60 50
```

```
** TEST INSERT (again) **
```

```
Inserting in this order: 20 40 10 70 99 -2 59 43
```

```
# of nodes: 11
# of leaves: 4
tree height: 4
tree empty? false
pre-order: 50 30 20 10 -2 40 43 60 59 70 99
in-order: -2 10 20 30 40 43 50 59 60 70 99
post-order: -2 10 20 43 40 30 59 99 70 60 50
```

```
*****
```

```
* test string BST *
*****
```

```
** CREATE BST **
```

```
# of nodes: 0
# of leaves: 0
tree height: -1
tree empty? true
```

Enter string file: /home/fac/mthayer/submit/23fq5005/files/p2-files/strings.dat



```

tree height: 2
tree empty?  false

** TEST TRAVERSALS **
pre-order:   mary gene bea jen sue pat uma
in-order:    bea gene jen mary pat sue uma
post-order:  bea jen gene pat uma sue mary

** TEST LEVEL & ANCESTORS **
level(mary): 0, ancestors(mary):
level(gene): 1, ancestors(gene): mary
level(bea): 2, ancestors(bea): gene mary
level(jen): 2, ancestors(jen): gene mary
level(sue): 1, ancestors(sue): mary
level(pat): 2, ancestors(pat): sue mary
level(uma): 2, ancestors(uma): sue mary

** TEST CONTAINS **
contains(gene): true
contains(mary): true
contains(bea): true
contains(uma): true
contains(yan): false
contains(amy): false
contains(ron): false
contains(opal): false

** TEST REMOVE **
removing in this order: gene mary bea uma yan amy ron opal
# of nodes: 3
# of leaves: 2
tree height: 1
tree empty?  false
pre-order:   pat jen sue
in-order:    jen pat sue
post-order:  jen sue pat

** TEST INSERT (again) **
inserting in this order: gene mary bea uma yan amy ron opal
# of nodes: 11
# of leaves: 4
tree height: 4
tree empty?  false
pre-order:   pat jen gene bea amy mary opal sue ron uma yan
in-order:    amy bea gene jen mary opal pat ron sue uma yan
post-order:  amy bea gene opal mary jen ron yan uma sue pat

Goodbye!

```

## Submission:

You must name your files **BST.h** and **p2.cpp**

To submit, type the following command at the prompt in the directory where the P2 files reside:

```
/home/fac/mthayer/submit/23fq5005/script/p2_runme
```

You have read/write permissions on your submission directory at:


```
/home/fac/mthayer/submit/23fq5005/p2/yourusername
```



(<https://seattleu.instructure.com/courses/1610311/modules/items/17916997>)

(<https://seattleu.instructure.com/courses/1610311/modules/items/17916997>)

0

[Note Oct 27 2021.pdf \(https://seattleu.instructure.com/courses/1610311/files/69761958?wrap=1\)](https://seattleu.instructure.com/courses/1610311/files/69761958?wrap=1)   
([https://seattleu.instructure.com/courses/1610311/files/69761958/download?download\\_frd=1](https://seattleu.instructure.com/courses/1610311/files/69761958/download?download_frd=1))


View Rubric

Select Grader


Elaine Huynh (TA) 



P2: BST		
Criteria	Ratings	Pts
Implement empty() <a href="#">view longer description</a>		3 / 3 pts
Implement size() <a href="#">view longer description</a>		10 / 10 pts
Implement getLeafCount() <a href="#">view longer description</a>		10 / 10 pts
Implement getHeight() <a href="#">view longer description</a>	<b>Comments</b> -3: result is incorrect due to error in code. Base cause should return -1, instead of 0	7 / 10 pts
Implement getLevel(T item) <a href="#">view longer description</a>		10 / 10 pts
Implement getAncestors(T item) <a href="#">view longer description</a>		10 / 10 pts
Implement copy constructor and overloaded assignment operator <a href="#">view longer description</a>		10 / 10 pts
Modify constructor, destructor, insert, contains, and remove <a href="#">view longer description</a>		10 / 10 pts
Modify traversals <a href="#">view longer description</a>		12 / 12 pts
Main client program <a href="#">view longer description</a>		10 / 10 pts



<https://seattleu.instructure.com/courses/1610311/modules/items/17916997>



<https://seattleu.instructure.com/courses/1610311/modules/items/17916997>

P2: BST		
Criteria	Ratings	Pts
	documentation in header files and above function prototypes for public and private methods. preconditions should document any restrictions on state before calling function; postconditions should document any changes to the object  You can make your string and int arrays variables that you pass as method parameters so you do not need to hardcode it multiple times:	
Other comments <a href="#">view longer description</a>	<b>Comments</b> Great job!	0 / 0 pts
		Total Points: 95.5