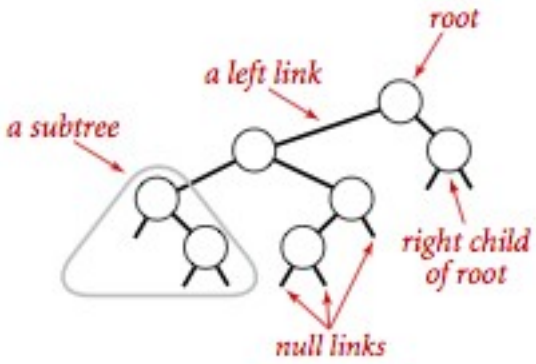
CS 360
Programming Assignment 4: Binary Tree

Date Assigned: Tuesday, May 29th, 2018
Date Due: Friday, June 8th, 2018 @ 1:00pm
Submit via Canvas and turn in a printout

Background

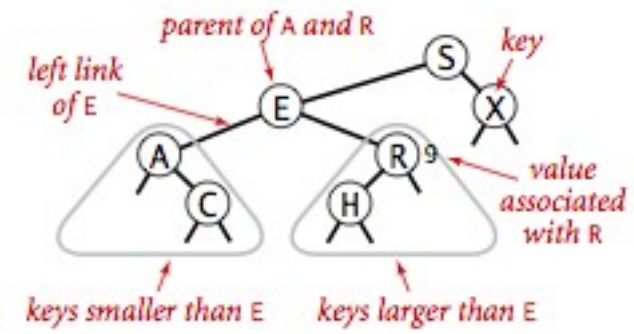
This assignment involves working with binary trees. Your'e welcome. Haha.

In a **binary tree**, every node is pointed to by just one other node, which is called the parent (except for a special node called the root), and each node has exactly two links, which are called the left and right links, which point to the nodes left and right children, respectively.



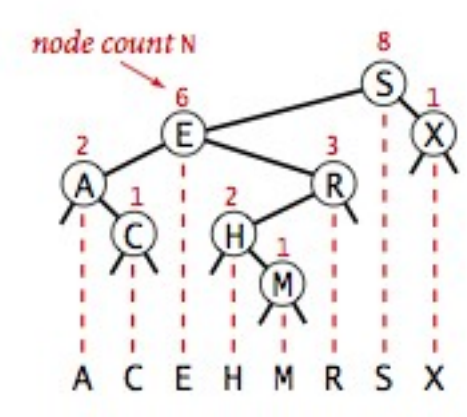
Anatomy of a binary tree

If we add a value to each node, then we have what is known as **binary search tree**. In a binary search tree, you load the tree by testing the value to be loaded against the value in each node. If the value to be loaded is less than the value of the node, then you go left. Otherwise, you go right. A binary search tree looks like this:



Anatomy of a binary search tree

The great thing about binary search trees, is that when you load the tree with a set of values, and then do an in-order traversal of the tree, printing the node contents as you go, you have an automagically ***sorted*** set of values. In the diagram below, when we look at the nodes from left to right order, they are already sorted!



The binary search tree data structure is one of the most fundamental structures in computer science.

In this assignment you will create the implementation of a simple Binary Tree in C++. This challenge will require a good understanding of pointers, objects and classes, along with the uses of "new" and "delete".

Getting Started

Download the following files from the "Assignments/Program4" folder on Canvas.

data.txt
Node.h
BinaryTree.h
main.cpp

The Task

Given the `Node.h`, and `BinaryTree.h` files above, create the implementation for the Node and BinaryTree classes. The `main.cpp` file is some starter code for your use if you want to use it.

Rules

1. You cannot modify `Node.h or BinaryTree.h`. This is important, because the main.cpp file that I will test your code with relies on the BinaryTree interface as specified in BinaryTree.h.

2. You can modify `main.cpp` as needed. It is simply code to get you started.
   - add print debug statements

- comment out functionality that you haven't yet implemented.
- etc.

3. You must clean up your memory when done, so you must provide a working destructor for BinaryTree.

4. The BinaryTree should not allow any duplicated values.

5. You must provide a working makefile that creates an executable file called “bst”.

Binary Tree Methods

`addNode(int value)` adds a new node to the tree in the proper location.

`search(int value)` returns a pointer to the node containing the value.

`cleanupTree()` cleans up the tree, freeing any memory that was allocated.

`printTree(bool ascending)` prints the tree in ascending order if ascending is true, and descending order otherwise.

Hints

- Work incrementally, starting with the easy stuff, and working your way up to the harder pieces. Note that you can comment out lines from `main.cpp` that you haven't got working yet.
  - Work on addNode() and printTree() first
- Think recursion! Recursion will be very helpful.
- Once you get the code working with the provided `data.txt` file, try modifying the contents of `data.txt` to have different values. Also, try your program with an empty `data.txt` file.
- Work on the BinaryTree destructor last....get everything else working and tested before attempting this one.
- Feel free to use your code from the “List” programming challenge. It could be helpful.
- Be smart about how you write your main.cpp file. If there is a method in the BinaryTree interface, you can bet that my main class will be testing it. So please make sure you write code to test each method thoroughly.

What to Submit

Your submission should include only the following files: **Node.cpp, BinaryTree.cpp** and **makefile.** You do not need to turn in main.cpp or the provided .h files.

Put copies of the above files into a folder with your name and “P4” in its name. Zip the folder, and submit the zip file to Canvas. You should also provide a printed copy.

Sample Output (using the provided data.txt):

Your output doesn't have to match mine, but getting it close to this might help you.

```
in BinaryTree constructor
adding value: 45
adding value: 22
       visiting node,left,right: 45,null,null
adding value: 99
       visiting node,left,right: 45,22,null
adding value: 55
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,null,null
adding value: 11
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,null,null
adding value: -5
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,null
       visiting node,left,right: 11,null,null
adding value: 76
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,55,null
       visiting node,left,right: 55,null,null
adding value: 78
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,55,null
       visiting node,left,right: 55,null,76
       visiting node,left,right: 76,null,null
adding value: 41
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,null
adding value: 9
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,41
       visiting node,left,right: 11,-5,null
       visiting node,left,right: -5,null,null
adding value: 1
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,41
       visiting node,left,right: 11,-5,null
       visiting node,left,right: -5,null,9
       visiting node,left,right: 9,null,null
adding value: 52
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,55,null
       visiting node,left,right: 55,null,76
adding value: 78
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,55,null
       visiting node,left,right: 55,52,76
       visiting node,left,right: 76,null,78
       visiting node,left,right: 78,null,null
adding value: 54
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,55,null
       visiting node,left,right: 55,52,76
```

```
       visiting node,left,right: 52,null,null
adding value: 8
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,41
       visiting node,left,right: 11,-5,null
       visiting node,left,right: -5,null,9
       visiting node,left,right: 9,1,null
       visiting node,left,right: 1,null,null
adding value: 3
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,41
       visiting node,left,right: 11,-5,null
       visiting node,left,right: -5,null,9
       visiting node,left,right: 9,1,null
       visiting node,left,right: 1,null,8
       visiting node,left,right: 8,null,null
adding value: 546
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,55,null
adding value: 23
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,41
       visiting node,left,right: 41,null,null
adding value: 71
       visiting node,left,right: 45,22,99
       visiting node,left,right: 99,55,546
       visiting node,left,right: 55,52,76
       visiting node,left,right: 76,null,78
adding value: 13
       visiting node,left,right: 45,22,99
       visiting node,left,right: 22,11,41
       visiting node,left,right: 11,-5,null
root: 45
printing tree ascending=========================
       val: -5
       val: 1
       val: 3
       val: 8
       val: 9
       val: 11
       val: 13
       val: 22
       val: 23
       val: 41
       val: 45
       val: 52
       val: 54
       val: 55
       val: 71
       val: 76
       val: 78
       val: 78
       val: 99
       val: 546
done printing tree.

BinaryTree::searching for 22
       BinaryTree::search(n,val): 45,22
       BinaryTree::search(n,val): 22,22
found it!

BinaryTree::searching for 26
       BinaryTree::search(n,val): 45,26
```

```
       BinaryTree::search(n,val): 22,26
       BinaryTree::search(n,val): 41,26
       BinaryTree::search(n,val): 23,26
value not found
in BinaryTree destructor
cleaning node: 45
cleaning node: 22
cleaning node: 11
cleaning node: -5
cleaning node: 9
cleaning node: 1
cleaning node: 8
cleaning node: 3
cleaning node: 13
cleaning node: 41
cleaning node: 23
cleaning node: 99
cleaning node: 55
cleaning node: 52
cleaning node: 54
cleaning node: 76
cleaning node: 71
cleaning node: 78
cleaning node: 78
cleaning node: 546
```

Run 2 (with an empty data.txt):

```
inside BinaryTree constructor
printing tree in ascending order=========================
done printing tree.
BinaryTree::searching for 1
tree empty, value not found
inside BinaryTree destructor
```