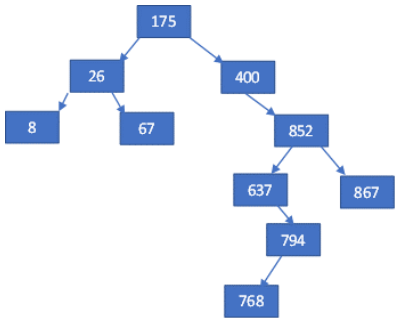
ETEC2101	Lab7: **(Binary Search) Trees**	Points: **100**

Tasks:

1. Basic functionality:
    a. (**10 points**) OrderedSet class, with hidden node class
        i. **Important**: to get full points, and unless otherwise specified, all methods in the OrderedSet class should be "stub" functions – meant mostly to call the recursive function in the node class.
        ii. Includes a default constructor and destructor
    b. (**5 points**) **insert** method
    c. (**12 points**) **traversal** method (which populates and returns an std::vector or ssuds::ArrayList) of all the values in the tree. Use an enum to indicate Pre-, Post-, or In-order. Make the stream operator output the result of an in-order traversal.
    d. (**14 points**) Have a **rebalance** method. This should create a new optimal root node and arrange children in an optimal manner. You *can* do this one in the OrderedSet class (as opposed to the node class). I used the traversal method results to help me here.
    e. (**5 points**) **clear** and **size** methods
    f. (**8 points**) bool **contains**(const T& val);
    g. (**8 points**) **get_height**() method. We could probably store this in the node, but I'd like you to calculate it on the fly. I disagree with Wikipedia a bit on this: I consider a 1-element tree to be height 1, not 0.
2. Advanced functionality (do enough of these to get your desired point value):
    a. (**15 points**) bool erase(const T& val) method – return true if it was removed (false if it wasn't there)
    b. (**20 points**) iterator that produces an in-order output of data *without caching / copying* the data in the set and that uses our Lab6 Stack class. I used https://www.geeksforgeeks.org/inorder-tree-traversal-without-recursion/ as a reference. You'll still need to do a bit of work to adapt this to an iterator, however.
    c. (**5 points**) The other 3 constructors and = operator (I had to add them anyway for some of my tests)
    d. (**18 points**) Make a **tree_string** method which returns this type of text-visualization of the tree




```
175
   L: 26
      L: 8
      R: 67
   R: 400
      R: 852
         L: 637
            R: 794
               L: 768
         R: 867
```

    e. (**10 points**) Include functions that overload the operators on the slides to do the set-operations (union, intersection, relative-difference)
3. (**15 points**) Google tests