Assignment Objectives:
- Implement program using queue, binary search tree, and AVL tree classes.
- Produce output to illustrate comparison between binary search tree and AVL tree balancing.

In this assignment, you will implement four class templates, a linked list, a queue, a binary search tree, and an AVL tree, including the standard functionality associated with those classes:

- The linked list requires (at minimum) a default constructor, a copy constructor, a destructor, and an insertion member function.
- The queue should be derived from the linked list, with (at minimum) a default constructor, a copy constructor, and enqueue and dequeue member functions.
- The binary search tree requires (at minimum) a default constructor, a copy constructor, a destructor, and an insertion member function, as well as several additional member functions that are less standard and customized for this program (described below).
- The AVL tree requires (at minimum) a default constructor, a copy constructor, a destructor, an insertion member function, and the various single and double rotation member functions that are needed to maintain the tree's balance (see Section 4.4 of the Weiss textbook for details). It will also require several additional member functions that are less standard and customized for this program (described below).

In addition to the data, left pointer, and right pointer fields, each tree node will require fields to hold the node's depth, the depth of node's parent, and the index of the node in an inorder traversal of the tree. These fields will be essential to perform the desired output of the tree, and recursive functions to traverse the tree's nodes and to populate these three fields for each node will be required for both types of tree.

The output operation itself should produce the values at each depth on a separate line, with slashes and dashes output to connect nodes, as well as asterisks to delineate the top and bottom borders of the nodes. The output member functions for each type of tree will accomplish this by utilizing a queue of subtrees, as outlined below:

- Enqueue the root of the entire tree.
- While the queue is not empty, dequeue the front element of the queue, and process its root, as described next.
- For all root values at a particular depth, start a countdown from 5 down to 0.
  - When the countdown value is 5, output the appropriately angled slash that would emerge from the bottom of the root's parent node, as the first part of the link between parent and offspring. Note that the inorder traversal index of a node indicates what “column” the node will occupy in the output.
  - When the countdown value is 4, output the appropriate number of dashes to horizontally connect the two slashes that start and end the link between the parent and its offspring.
  - When the countdown value is 3, output the appropriately angled slash that would emerge from the top of the root's node, as the last part of its link to its parent node.
- When the countdown value is 2, output the appropriate number of asterisks to serve as a “cap” over the root node’s data value.
- When the countdown value is 1, output the root node’s data value.
- Finally, when the countdown value is 0, output the appropriate number of asterisks to serve as a “base” underneath the root node’s data value.
- For countdown values 1 through 5, enqueue the root back into the queue after processing, but for countdown value 0, enqueue the root’s offspring instead.

- Keep track of the inorder traversal number of the first node at each depth. When it reappears, decrement the countdown. When the countdown reaches 0, proceed in this manner with the next depth level, until the queue has been completely emptied (i.e., the tree has been completely output).

Write a driver program that will load both trees with a user-specified set of string data (several examples are provided on the course website), and output both trees to a user-specified output file.

An example is illustrated below:

```
Jacob
Isabella
Ethan
Emma
Michael
Olivia
Alexander
Sophia
William
Ava
Joshua
Emily
Daniel
Madison
Jayden
Abigail
Noah
Chloe
Anthony
Mia
```

Zip-compress your entire program folder and place it on your dropbox by the deadline mentioned above. Several files of test data are available on the course website.

You must write your own code on this assignment, with adequate explanatory documentation (i.e., a paragraph at the top of each program file, including your name and a description of the file's contents, and a sentence preceding each function, describing the function's purpose).

Obtaining code assistance from any outside source is considered academic misconduct.

The only person permitted to see your code prior to the assignment deadline is the instructor.