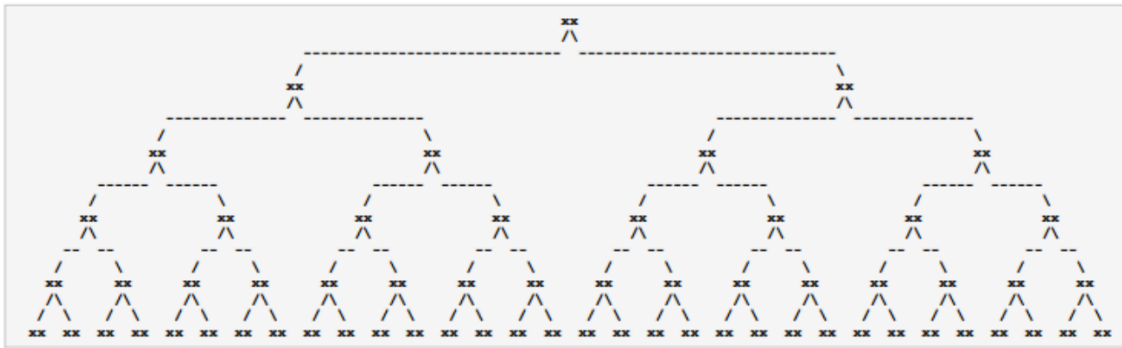


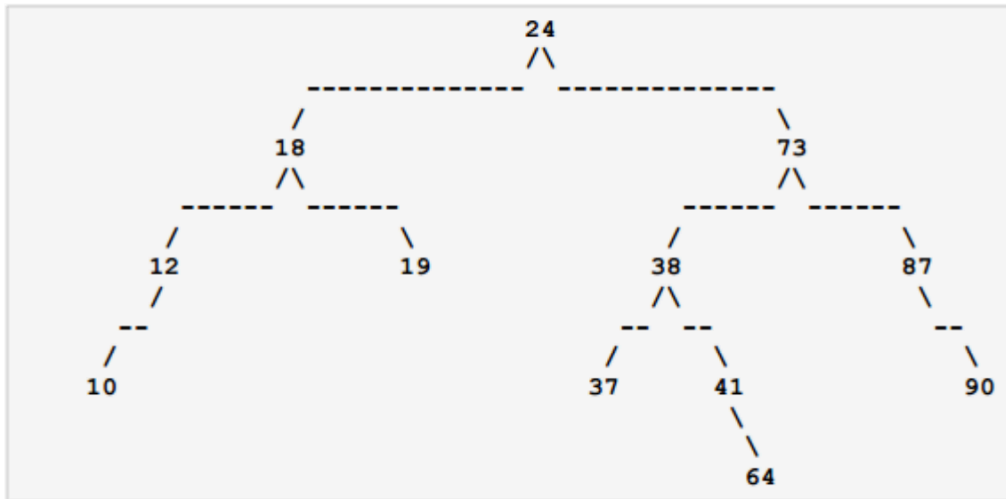
Project 2

A tree printer

You are provided a **TreePrinter** class that has a **print()** method that will print any arbitrary binary tree. A template for how it prints a tree:



TreePrinter is able to print trees with height up to 5, i.e., 32 node values on the bottom row. An example of an actual printed tree:



Part 1

The first part of the assignment makes sure that you can successfully insert nodes into, and delete nodes from, a BST and an AVL tree.

First, generate a random BST that has height 5 and contains random values from 10 through 99. You may have to generate dozens of trees until you get one that's exactly height 5. Don't worry that the tree is unbalanced. Print the tree.

Now repeatedly delete the root of the tree. Print the tree after each deletion to verify that you did the deletion correctly. Stop when the tree becomes empty.

Second, create an AVL tree node by node. Generate 35 unique random integers 10-99 to insert into the tree. Print the tree after each insertion to verify that you are keeping it balanced. Each time you do a rebalancing, print a message indicating which rotation operation and which node. For example:

Double left-right rotation: 76

As you did with the BST, repeatedly delete the root of your AVL tree. Print the tree after each deletion to verify that you are keeping it balanced.

A handy AVL tree balance checker is included in your lecture slides for AVL Trees.

Part 2

The second part of the assignment compares the performance of a BST vs. an AVL tree. First, generate n random integers. n is some large number, explained below. Time and print how long it takes to insert the random integers one at a time into an initially empty BST. Do not print the tree after each insertion.

Time and print how long it takes to insert the same random integers one at a time into an initially empty AVL tree. Again, do not print the tree after each insertion.

Code

You can use any code from the lectures or from the textbook. You do not have to use parameterized generic types. You can use raw (nongeneric) types, or `<Integer>`

Teamwork

You may work individually as a team of one, or you can partner with another student as a team of two.

You can be on only one team at a time. If you partner with someone, both of you will receive the same score for this assignment. You'll be able to choose a different partner or work alone for subsequent assignments.

What to turn in

Create a zip file containing:

- Your Java source files.
- Any instructions on how to build and run your code.
- Text files containing your outputs
- A short report (1 or 2 pages) that describes your conclusions from doing this assignment.

Name the zip file after yourself or yourselves.

Examples: smith.zip, smith-jones.zip