CSCI 220 -- PA 11

AVL Trees

Feel free to discuss and help each other out but does not imply that you can give away your code or your answers! Make sure to read all instructions before attempting this lab.

You can work with a lab partner and each one must submit the same PDF file (include both names in the submission file). Each person must include a brief statement about your contribution to this assignment.

You must use an appropriate provided template from Canvas or my website (zeus.mtsac.edu/~tvo) and output "Author: Your Name(s)" for all your programs. If you are modifying an existing program, use "Modified by: Your Name(s)".

Exercise 1: Use **AVLTree** class in C++ book (modified by me and provided here) or **AVLTreeMap** class for Java book and set up a test driver to perform some operations such as insert, erase, and find. Perform the operations in question 1 below (steps 1 to 7) and then search for 15, 30, and 8. Print the AVL tree as the final step (include key, value, and height of each node). Assume that key is an integer and value is a string such as a name (come up with your own names).

Exercise 2: You will implement a better population database for California counties using an AVL tree from exercise 1 to store the database records. Define and implement **PopBetterMap** class that supports standard map operations using county code as a key for each record (no duplicate keys). Your **PopBetterMap** class uses an AVL tree to store population records. Download the data file ***p4small.txt***, containing a list of a few population records – county code, population in million, and county with state abbreviation (3 fields separated by commas). Build the AVL tree from the records of the input data file by inserting one record at a time to the tree. Run the following test cases after the tree is constructed:

1. List all records
2. Search for 6037
3. Search for 6000
4. Insert 6066, 1, “New County, CA”
5. Insert 6065, 2000, “Riverside, CA”
6. Delete 6999
7. Delete 6075
8. Delete 6055
9. List all records

```
Class PopBetterMap
  // private data
      // set up AVLTree or AVLTreeMap

  // public operations
      // constructor accepts file name and construct AVL tree
      PopBetterMap(string filename)

      // print appropriate message and data if found
      void find(int code)

      // print appropriate message and insert node if not found
      // replace data if found
      void insert(int code, int pop, string county)

      // print appropriate message and erase node if found
      void erase(int code)

      // print one record per line using an in-order traversal;
      // include height of each node
      void print()
```

Question 1: Provide the final AVL tree after the following operations (showing keys only). You might want to draw multiple trees along the way to show your work.

1. Insert 10
2. Insert 20
3. Insert 4
4. Insert 8
5. Insert 15
6. Erase 8
7. Erase 10

Question 1: Provide the AVL tree after the following operations (showing keys only). You might want to draw multiple trees along the way to show your work.

Insert 10, Insert 20, Insert 30, Insert 15, Insert 12, Insert 20, Erase 30

Question 2: What is a splay tree? Given a binary search tree class and an AVL tree class, what is the best to implement a splay tree (i.e., use inheritance or composition and which one)?

Extra Credit: Add operation draw() to PopBetterMap class to draw the tree (key only). You can submit one version here to include exercise 2. See sample drawing below where

1234 is the root with left child 1000. 2000 is the right child of the root with 2 children, 1500 and 2000.

```
1234
    1000
    2000
        1500
        2000
```

Fill out and turn in the PA submission file for this assignment (save as PDF format).