CS-202 Data Structures

Assignment 3

AVL Trees

Due Date: Monday 8th November, 11:55pm

Late Submission Policy

20% per day deduction for up to 2 days after the due date

Part 1
AVL Trees

In this part you'll implement an AVL tree. Apart from the functions implemented for BSTs in the previous assignment, you need to implement the following additional functions:

- node* rotateleft(node* p): p is root of the tree before rotation, the function should return pointer to the new root of the tree, around which the subtrees were rotated. Remember to fix the heights of the subtrees.
- node* rotateright(node* p): Same as above.
- int balanceFactor(node* p): Calculates and returns the balance factor of the node p based upon the heights of left and right subtrees.
- int height(node* p): Calculates height of the tree rooted at p

Note: For the other functions, you may use the code you wrote for implementing a BST in the previous assignment, however, you would need to make slight modifications to balance the BST. Feel free to add any helper functions you like.

Please ensure your code compiles properly on a Linux environment.

You may use test1.cpp to verify the correctness of your code.

PART 2
Index

In this part you'll write a cross-reference program which constructs an index object using an AVL tree with all words included from a text file. You will record the line numbers on which these words occur by storing these numbers on linked lists associated with the nodes of the tree. After the input file has been processed, you have to generate an output file which displays all the words of the text file in alphabetical order along with the corresponding list of line numbers in ascending order.

Your index should also support searching for an entry, deleting an entry, updating an entry and inserting a new entry.

Example Output:

alpha 1, 3, 55, 67, 98

beta 34, 54, 67, 109

gamma 123, 456, 678

You need to implement the following functions:

- Index(): Creates an empty (i.e. without words) Index Object.
- void initialize(string wordsFile): Populates words read from wordsFile into wordsTree, **wordsFile is the name of the file**.
- node* findWord(string word): Corresponds to node* searchNode(T k, S val).
- void deleteWord(string word): Corresponds to void deleteNode(T k, S val).

- void editWord(string oldWord, string newWord): Changes the key.
- void insertWord(string word): Corresponds to void insertNode(node* p).

During file reading, make sure you convert all words to lowercase (you are free to use libraries for this) and store words with punctuations removed. Feel free to add any helper functions you like. You are allowed to make modifications to the LinkedList file.

Note: You will be using your implementation of AVL trees from part 1 of this assignment. Make sure you go through the README file before starting this part.

Marks Distribution

Part 1: AVL Tree	**30**
Part 2: Index	70

Submission Policy:

* ZIP all the files and name the folder as PA3_rollnumber.zip

*Start the assignment early!

Happy Coding:)