CSCI 235 - Spring 2015
Homework 5

100 points

Due: Sunday, May 24th, 2015, noon

1 Word Occurrence Index

An important document is given to you and your job is to create an online index of it so as to be able to find any word's location in the text quickly. You have decided that the best way to do this is to create a binary search tree, with each node containing the word you are searching for along with a linked list (or queue) that contains the line number of the transcript where the word occurs. This way, you can know at what line(s) in the text a word occurs. You must make sure that your program filters the words on input to treat upper and lower case as the same, remove punctuation marks, and not store numerical data as words. Read the text in from a data file specified by the user, and build a binary search tree index.

1. Create the index (tree) for the document.

2. Print the index out in lexicographic order with a count of the number of times the word occurred and a list of each line number that the word is found on. Use an inorder traversal to print the tree out. Below is a partial sample output for the document provided on Blackboard:

```
a :: Count: 89 :: Lines: [1, 1, 1, 3, 3, 5, 5, 7, 11, 13, 13, 15, 19,
19, 19, 19, 24, 24, 26, 26, 28, 28, 30, 34, 36, 38, 42, 42, 42, 48,
48, 50, 54, 54, 54, 56, 56, 56, 56, 56, 58, 58, 70, 72, 75, 75, 75,
75, 77, 77, 77, 79, 79, 79, 83, 83, 83, 85, 85, 87, 91, 91, 93, 93,
95, 95, 99, 101, 105, 105, 105, 109, 111, 111, 111, 111, 111, 113, 115,
117, 117, 119, 119, 121, 123, 125, 125, 131, 131]
able :: Count: 2 :: Lines: [3, 77]
about :: Count: 7 :: Lines: [13, 19, 26, 28, 30, 75, 129]
academic :: Count: 1 :: Lines: [70]
accessory :: Count: 1 :: Lines: [99]
according :: Count: 1 :: Lines: [77]
accuracy :: Count: 1 :: Lines: [89]
across :: Count: 1 :: Lines: [77]
```

```
add :: Count: 1 :: Lines: [101]
adding :: Count: 1 :: Lines: [3]
adopt :: Count: 1 :: Lines: [105]
advantages :: Count: 1 :: Lines: [64]
advertising :: Count: 1 :: Lines: [50]
advisory :: Count: 1 :: Lines: [93]
after :: Count: 2 :: Lines: [42, 56]
afternoon :: Count: 1 :: Lines: [85]
afterthought :: Count: 1 :: Lines: [44]
again :: Count: 2 :: Lines: [5, 121]
aggressively :: Count: 1 :: Lines: [24]
ago :: Count: 3 :: Lines: [19, 40, 54]
...
```

3. Print the total number of tree nodes.

4. Print out the height of the tree.

5. Print out the word with the maximum number of occurrences and on what line(s) this word occurs.

6. Interactively request the user to input a word and return the number of occurrences of that word and the line numbers they occur on. A message should be printed if the word is not in the index.

For testing purposes, create a small document yourself, or use the first 3 - 4 lines of the given document.

1.1 Extra Credit (50 points)

1. Implement the deletion of a given word. In order to test, ask the user to provide a word, and delete it. Then search for the same word again and demonstrate that it is not in the tree.

2. Display the tree in a level-order traversal.

1.2 Programming Tips

Note: Use as many functions as you need, i.e. you don't have to implement all tree functions if you don't need them. However, you should provide at least a copy-constructor, an assignment operator and a destructor for the tree.

Start from the partial tree implementation that will be provided on blackboard. Note that the BinaryNode is template of two types. So a given BinaryNode for the tree that stores a string (this is the key) and a LinkedQueue of integers (you can also use a LinkedList of integers) can be syntactically defined as follows:

```
BinaryNode< string, LinkedQueue<int> > a_binary_node;
```

The binary search tree you will create will be defined in your main file as:
```
BinarySearchTree<string, LinkedQueue<int> > binary_tree;
```
or
```
BinarySearchTree<string, LinkedList<int> > binary_tree;
```

**However, first get the tree to work with simple data types before working
with lists or queues.**

2   Submission

Once you have ensured that your program works on the Linux machines in Lab 1000G in
Hunter North, pack all files in a zip/tar.gz archive and name it `<YOUR FULL NAME>_hw5.zip`
or `<YOUR FULL NAME>_hw5.tar.gz`. Replace `<YOUR FULL NAME>` with your full name, us-
ing underscores for spaces. Submit this file on Blackboard in the section for Homework 5.
Please make sure you have included the Makefile and that your program compiles using
the `make all` command. Your program will not be graded at all if any of the following is
true:

1. The files were not archived (zipped)

2. There is no Makefile

3. There is a compile error

4. It's missing any of the files required by this assignment

5. It crashes giving a segfault error.

Please read the Programming Rules document on Blackboard. Good luck!