Programming HW4
CSCI 235
Ilya Korsunsky
Due Friday 5/15/15

Programming Problem: 100 points + 15 EC points

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 called *document1.txt* in the blackboard, and build a binary search tree index.

(a) Create the index for the document.

(b) 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 i*norder traversal* to print the tree out. Below is a partial sample output:

```
a  Count: 46 Lines:  12,14,16,17,17,22,23,24,26,28,
32,35,36,37,38,39,42,47,52,53,57,
60,78,81,88,88,90,91,97,97,116,119,
132,136,151,153,153,165,174,175,176,177,181,
184,189,190,
able  Count:  1 Lines: 32,
about  Count:  1 Lines: 192,
access  Count:  1 Lines: 150,
according  Count:  1 Lines: 77,
acknowledged  Count:  1 Lines: 70,
act  Count:  1 Lines: 29,
activity  Count:  1 Lines: 133,
advantage  Count:  1 Lines: 160,
affirmative  Count:  3 Lines: 71,84,96,
after  Count:  1 Lines: 102,
afterward  Count:  1 Lines: 102,
all  Count:  3 Lines: 54,181,192,
allegations  Count:  1 Lines: 85,
also  Count:  5 Lines: 70,77,90,126,133,
```

```
although  Count:  1 Lines: 131,
among  Count:  3 Lines: 114,151,151,
amount  Count:  2 Lines: 81,117,
.       .     .   .    .
.       .     .   .    .
```

(c) Print the total number of tree nodes.
(d) Print out the height of the tree.
(e) Print out the word with the maximum number of occurrences and on what line(s) this word occurs.
(f) 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 purpose, you can use the file *document1_short.txt* that is a shorter version of the main file

EXTRA CREDIT (15 points)

(g) 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.

(h) Display the tree in a level-order traversal.

PROGRAMMING TIPS:

Start from the partial tree implementation that will be provided on blackboard. Note that the BinaryNode is template given 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;

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