CS212 Data Structures and Algorithms
Program 2 – Babble

In this program, you will write code to read a text file and compute word co-occurrence statistics. You'll then use the program to read a text file, such as a book by Dickens, a book of the King James Bible, or a reading assignment, and generate some meaningless babble that superficially resembles the source file. In fact, the higher-order the statistics, the more seemingly sensible it will be, until you study it closely. You'll be given some sample C# code to start from that reads, tokenizes, and displays a text file. You have two options.

The first option, worth a maximum of 90% credit, computes first-order statistics. That is, for each word, it finds all the words that follow it in the file. Then, when generating the output, it picks a new word at random from that list. That word it chose is added to the output and also used as the next key to generate the next word.

To do this, use a hash table (Dictionary) of ArrayList objects, defined as in the HashExample code. Every time a pair of words is read from the file, retrieve the appropriate ArrayList of succeeding words by using the first word as the key in the hash table. Then Add the second word onto the ArrayList. To generate output, given the previous word, find the appropriate list of following words in the hash table and pick a word at random from that list. Initially use the first word of the file. Note that if you happen to end up with the last word in the source text, there may not be a successor, so you can either start again at the beginning of the text or pick a spot to restart at random.

The second option, for full credit, computes Nth-order statistics. N will be set by a UI widget; possible values are 1 to 5. If N is 4 and the words "It was the best" occur 20 times in the input file, when you are generating text, the next word you generate should be one of the 20 successors, selected at random. Your key for the hash table should be a combination of the prior N words, e.g. It-was-the-best, and the list of things stored at that key the next words.

When the Load button is clicked, the program should read the selected file, analyze it according to the currently selected order, and display the total number of words and the number of unique keys found (for first-order statistics, the number of unique words; for Nth-order statistics, the number of unique sequences of N words). When the order is changed, statistics should be recomputed for the new order. When the "Babble" button is clicked, the program should clear the output and generate 200 words of random text based on the computed statistics.

Make sure to comment your functions, describing what they do and input and output conditions. You should use other comments in the code where it would help make clear what is going on. Use meaningful variable and function names and good programming style.

To get started with this project, open a new WPF project in Visual Studio. (Don't use Console App or Windows Forms or other project types.) Delete the C# source code in the .cs file and paste in the contents of BabbleSample.cs. Delete the contents of the .xaml file and paste in BabbleSample.xaml. Then build and run.

Also zip up your project directory and submit it through moodle. In the comments box note whether you got all parts working, first-order vs nth-order, etc.

CS212 Program 2 – Grading Sheet
Due date: ____________

Name: ___________________________ Section: __ Date turned in: _______ Late? _____

Parts of the program I didn't get to work:

Comments on this assignment:

____________________________For Grader's Use______________________________

- Program compiles and runs (40) _____

- Reads file; displays number of words and number of unique words (10) _____

- Computes statistics and generates appropriate random text (20) _____

- Algorithm behaves as described. Uses hash tables appropriately (10) _____

- Uses Nth-order statistics (1-5) (10) _____

- Style and mechanics: good programming style including comments (10) _____

Total (100) _____