**Lab 7**

This week you will write a program that will count the frequency of words in a text file. As you come across new words, store them in a dynamic array. If you find already existing words, increment the frequency of that word.

**Dynamic Arrays**

Dynamic arrays are just like regular arrays except they are declared dynamically through a pointer. This allows them to be resized later. To compare:

int myArray[10]; // regular array

int\* myArray = new int[10]; // dynamic array, same size as above

If we fill up this 10 element array, then we’ll need to resize it. First, create a temporary pointer that will hold the new array. Set that equal to a new array that is a larger size (for example, the original size + 10, or \* 2). Then, copy over each element from the old array into the new array. Third, delete the old array. Finally, set the pointer of the old array back to the new array. This will let the variable name stay the same but contain the new, resized array.

**Structures**

You are to have a class called Word. The Word class should contain:

* A private string to store the word
* A private integer to store the frequency of occurrences of the word
* Public function increment() to increment the counter.
* Public functions getWord() and getCount() to return those data elements.
* Public functions setWord() and setCount() to set the values of those data elements.
* Since it’s an array, you’ll need a default constructor. This should set the word and frequency to default values.
* A constructor that takes in the word. This constructor will build the Word object to store that word internally, and set its initial count. (This could be 0 or 1 depending on how to you choose to write it.)
* Place the class declaration in Word.h and all function definitions in Word.cpp file.

**Assignment**

Your main function should be located in Lab7.cpp. In this file, you will need a dynamic array of Word objects. The array should start at 10 and double each time you resize. Remember that each time you have to copy the data inside of each object from the old array to the new one as you do this.

You should have a function called resizeArray(). This function should take two parameters: A reference to the array pointer, and the size of the array. (Since it’s a pointer, we just pass it as a pointer. However, since arrays are passed by reference in functions, we have to include both the pointer and the reference.) Since the function knows the array size, you know how big to make the new array. This function can either modify the size directly (pass by reference) or return the new size (which main() will have to capture).

Our arrays have needed a variable to keep track of how full it is, however, with dynamic arrays you also need to keep track of the size, since this is now variable. When do you resize? When the number of elements in the array matches the size of the array. (Remember, this is where good variable names come in handy.)

**File Input/Output**

File input will consist of a file called input.txt that has a bunch of words. File output will consist of a file called output.txt. The first line should say “Words found: “ along with a space, then the number of words found. The second line should say “Array’s max size: “ followed by a space and the largest size the array got to. After a blank line, each word should be listed, followed by a space, a hyphen, and a space, along with the frequency the word appeared. See the sample output on Blackboard for an example.

Note: using the stream operators on the file like we have been results in some weird combinations of “words.” This is ok. Your output file should still match the sample one posted.

**Grading and Turning In**

Turn in your program before the end of the lab. Only upload the .cpp and .h files. Grading will be performed using an automated grading program with a different input file (still input.txt) and will be compared to an output file (output.txt). The lab instructor will still manually look at your code for the other grading criteria (functional composition, style, etc.)