CSCI 2270 – Data structures and algorithms
Instructor: Hoenigman
Assignment 2
Due Friday, January 30 before 3pm

Word analysis

There are several fields in computer science that aim to understand how people use language. This can include analyzing the most frequently used words by certain authors, and then going one step further to ask a question such as: “Given what we know about Hemingway’s language patterns, do we believe Hemingway wrote this lost manuscript?” In this assignment, we’re going to do a basic introduction to document analysis by determining the number of unique words and the most frequently used words in two documents.

Please read all directions for the assignment carefully. This write-up contains both the details of what your program needs to do as well as implementation requirements for how the functionality needs to be implemented.

What your program needs to do

There are two test files on Moodle – *HungerGames_edit.txt* and *Hemingway_edit.txt* that contain the full text from *Hunger Games Book 1* and *Old Man and the Sea*, respectively. We have pre-processed the files to remove all punctuation and down-cased all words.

Your program needs to read in a .txt file, with the name of the file to open set as a command-line argument, and output the top *n* words (*n* is also a command-line argument) and the number of times the word was found, the number of array doublings needed to store the words found, and the total number of unique words in the file. In determining word frequency, exclude the 50-most-common words listed in Table 1.

Example:

Running your program using:

```
./Assignment2 Hemingway_edit.txt 10
```

would return the 10 most common words in the file *Hemingway_edit.txt* and should produce the following results.

```
441 - was
271 - fish
249 - old
246 - man
230 - him
205 - had
```

```
191 - is
189 - said
173 - now
165 - thought
#
Array doubled: 5
#
Unique non-common words: 2585
#
Total non-common words: 15186
```

Program specifications

Use an array of structs to store the words and their counts

There are specific requirements for how your program needs to be implemented. For this assignment, you need to use an **array of structs** to store the words and their counts. The members of the struct are left to you, but keep it as simple as possible.

Use two command-line arguments

Your program needs to have two command-line arguments – the first argument is the name of the file to open and read, and the second argument is the number of most frequent words to output. For example, running

```
./Assignment2 Hemingway_edit.txt 20
```

will read the *Hemingway_edit.txt* file and output the *20* most common words found in the file.

Use the array-doubling algorithm to increase the size of your array

We don't know ahead of time how many unique words either of these files has, so you don't know how big the array should be. Start with an array size of 100, and double the size as words are read in from the file and the array fills up with new words. Use dynamic memory allocation to create your array, copy the values from the current array into the new array, and then free the memory used for the current array. This is the same process you used in Recitation 2.

Format your output the following way

When you output the top n words in the file, the output needs to be in order, with the most frequent word printed first. The format for the output needs to be:

Count - Word
#
Array doubled: <number of array doublings>
#
Unique non-common words: <number of unique words>
#

Total non-common words: <total number of words>

Generate the output with these commands:

```
cout<<numCount<<” – “<<word<<endl;
cout<<”#”<<endl;
cout<<”Array doubled: “<<numDoublings<<endl;
cout<<”#”<<endl;
cout<<”Unique non-common words: “<<numUniqueWords<<endl;
```

Exclude these top 50 common words from your word counting
Table 1 shows the 50 most common words in the English language. In your code, exclude these words from the words you count in the .txt file. Your code should include a separate function to determine if the current word read from the .txt file is on this list and only process the word if it is not.

Table 1. Top 50 most common words in the English language

Rank	Word	Rank	Word	Rank	Word
1	The	18	You	35	One
2	Be	19	Do	36	All
3	Two	20	At	37	Would
4	Of	21	This	38	There
5	And	22	But	39	Their
6	A	23	His	40	What
7	In	24	By	41	So
8	that	25	From	42	Up
9	Have	26	They	43	Out
10	I	27	We	44	If
11	It	28	Say	45	About
12	For	29	Her	46	Who
13	not	30	She	47	Get
14	On	31	Or	48	Which
15	With	32	An	49	Go
16	He	33	Will	50	Me
17	As	34	My		

Output the top n most frequent words
Write a function to determine the top n words in the array. This can be a function that sorts the entire array, or a function that generates an array of the *n* top items. Output the n most frequent words in the order of most frequent to least frequent.

Submitting Your Code:
Submit your assignment to the COG autograder: https://web-cog.cs.colorado.edu/submit.html.

Login to COG using your identikey and password. Select the CSCI2270 - Hoenigman – HW #02 from the dropdown. Upload your file and click Submit. **Your file needs to be named Assignment2.cpp for the grading script to run.** COG will run its tests and display the results in the window below the Submit button. If your code doesn't run correctly on COG, read the error messages carefully, correct the mistakes in your code, and upload a new file. You can modify your code and resubmit as many times as you need to, up until the assignment due date.

In addition to submitting through COG, submit your .cpp file through Moodle using the Assignment 2 Submit link. Make sure your code is commented enough to describe what it is doing. Include a comment block at the top of the .cpp file with your name, assignment number, and course instructor.

If you do not get your assignment to run on COG, you will have the option of scheduling an interview grade with your TA to get a grade for the assignment. Even if you do get the assignment to run on COG, you can schedule the interview if you just want to talk about the assignment and get feedback on your implementation.

What to do if you have questions

There are several ways to get help on assignments in 2270, and depending on your question, some sources are better than others. There is a Peer Discussion Forum on our Moodle page that is a good place to post technical questions, such as how to shift an array. When you answer other students' questions on the forum, please do not post entire assignment solutions. The multi-course LAs are also a good source of technical information, especially questions about C++. If, after reading the assignment write-up, you need clarification on what you're being asked to do in the assignment, the TAs and the Instructor are better sources of information than the discussion forum or the LAs.