Problem Description The word count is the number of words in a document or passage of text. Word counting may be needed when a text is required to stay within specific numbers of words. This may particularly be the case in academia, legal proceedings, journalism, and advertising. Word count is commonly used by translators to determine the price of a translation job. Word counts may also be used to calculate measures of readability and to measure typing and reading speeds (usually in words per minute). When converting character counts to words, a measure of 5 or 6 characters to a word is generally used for English.

We will be doing a version of map-reduce using MPI to perform word counting over a large number of files.
There are three steps for this process:

- the MASTER node reads the file list (or a directory), which will contain the names of all the files that are to be counted. Note that only 1 of your processes should read the files list. Then each of the processes should receive their portion of the file from the MASTER process. Once a process has received its list of files to process, it should then read in each of the files and perform a word counting, keeping track of the frequency each word found in the files occurs. We will call the histogram produced the local histogram. This is similar to the map stage or map-reduce.
- the second phase is combining the frequencies of words across processes. For example, the word 'cat' might be counted in multiple processes, and we need to add up all these occurrences. This is similar to the reduced stage of map-reduce.
- the last phase is to have each of the processes send their local histograms to the master process. The MASTER process just needs to gather up all this information. Note that there will be duplicate words between processes. The master should create a CSV formatted file with the words and frequencies ordered.

Notes
The hard part of the problem concerns to split equally the computation among the processors. For instance, if we split the files between processors, we cannot have a good partitioning scheme because some files must be bigger and bigger than other files. A good partitioning scheme must consider the number of words in each file, and split according to this value.