Assignment 02: Building an MPI based text frequency counting system.

In this assignment you will be building a text frequency searching system. The program will read in a text file and will split it amongst all nodes that are taking part in the system. each node is responsible for taking its partition of text and finding how often each word occurs in it. This information will then be passed back to the master which will aggregate the results. This is a task that many search engines like google and yahoo would use to determine which words present the most information in a document and those words that determine the least information. Your program must be able to read a standard text file

To clarify if I have a text file with these contents:

```
the the the the

the are are the

big big are the

ale cow pig bum
```

The result of your program should be

```
the 7

are 3

big 2

ale 1

cow 1

pig 1

bum 1
```

note there are no repeating words. this should be your final result if you manage to implement 07. if you do not implement this you will have repeating counts but that is enough to get you 75% of this assignment.

Tasks and marking scheme:

01) create a main method that will initialise and finalise MPI, and will determine the world rank and world size. if the rank is zero the process should be the coordinator otherwise a participant (5%)

02) create a coordinator(int world_rank, int world_size) function that does the following tasks (30%)

- determine if the file exists broadcase a zero if the file exists and a -1 if not
- read the file into a char array or std::string determine it's length too
  - if using the char array fopen, fclose, fseek, ftell and fread will be useful
- adjust each offset to fall on a non alphabetic character
- determine the length of each partition to send use a scatter to send this length to

each participant

- use a scatterv to send partitions to each partition.
- calculate the frequency of each word and display the results on the console

03) create a particpant(int world_rank, int world_size) function that does the following tasks (20%)

- waits for a broadcast message from the coordinator if the message is -1 then the function exits immediately.
- takes part in a scatter to get the length of its partition and allocates the necessary memory
- takes part in a scatterv to get the partition data
- calculates the frequency of each word
- displays those results on the console

04) create an adjustStarts(char *data, int *offsets, int world_size) function that will adjust the offsets such that each offset falls on a character that is outside of the ranges [a,z] and [A,Z] (5%)

05) create an incrementWord(std::vector<std::string>& words, std::vector<int>& counts, std::string word) function that will increment the freqency of word if it exists in the words vector. if the word does not exist in the vector then append it to words and append a new count onto the counts vector. (5%)

06) create a getFrequencies(std::string& text, std::vector<std::string>& words, std::vector<int> counts) function that will go through the variable text and will pick out and count the occurence of all words in that text. it will store the words in the words vector and the associated counts in the counts vector. A word for this assignment is defined as a mix of uppercase and lowercase letters with no other characters in between (10%)

07) modify your code to send the list of counts and words back to the coordinator. the coordinator should aggregate this list and remove all duplicates. (25%)

Notes:

You have three weeks to submit this assignment. Submission date is 13th April at 23:55. you must submit via moodle. assignments that are submitted from 23:56 onwards will have the standard late penalties applied. You are required to submit a single CPP file that contains the entire code of your program

There are a range of penalties in addition to the standard late submission penalties that I can apply to assignment. This is to encourage you to submit your project in a form that makes it easy to run and correct.

- failure of code to compile -30% Using a standard makefile or compiler I should only have to link your code against the MPI libraries and it should run without modification
- the use of external libraries to MPI -20%
- filename name should be firstname_lastname_studentnumber where firstname, lastname and studentnumber are your own details.
- archives should only be in one of the following formats zip/rar/tar.bz2/tar.gz/7z -10%