Assignment 2
Basics of parallel programming with MPI
Due date: Monday, October 31, 11:59pm

Assignment description
In this assignment you will have to implement a simple parallelized program that calculates and visualizes the character count for all letters in a text file using MPI. Your program will have to do the following:

- Read the file into an array of chars.
- Go through every character in the array and calculate the number of occurrences of every letter. These counts are known as histograms. You can assume that the text is written using the 26-letter English alphabet. All other characters must be ignored. Upper- and lower-case forms of the same letter must be treated the same - e.g. 'A' counts as an occurrence of 'a'.
- Your calculations **must** use MPI. The character array will be loaded on the main process (pid=0) and then shared with the rest of the processes. How you do this is up to you.
  Each process will compute the character count for a section of the original char array. Once all processes have finished calculating, their character counts are merged using MPI on the main process (PID=0).
- Be careful with the character array size - **do not** hardcode it! It must be determined at runtime and you must malloc the array in the main process and the necessary arrays in the additional worker processes. Remember to free all data array when you are done.
- Once you have processed the entire array, your program would display the character counts in the main process in one of the two ways:

  1. Simple number counts, e.g.:
     ```
     a 10551
     b 1835
     c 3072
     d 6150
     e 1705
     ...
     ```
  2. Letter count histogram, where the length of each bar - i.e. the number of characters in it - is based the calculated count for that character. For example, if there were 10 a's, 8 b's, and 13 c's, the first three lines of your program's output should look like this:
     ```
     aaaaaaaaaa
     bbbbbbbb
     ccccccccccccc
     ...
     ```
     However, our character counts are often huge, so the histogram has to be appropriately scaled. The screen is often assumed to be only 80 characters wide, but many text files

will have histogram bars more than 80 characters long, resulting in rather ugly output. To avoid this, you will have to scale the length of the histogram bars, so that the longest bar is no more than 40 characters long.

- If the program received the appropriate command line argument, it will also save the histogram to the file. The saved version must display one number per line - counts for "a" on line 1, for "b" - on line 2, etc.. Do not include the letter itself. Sample file will look something like this:

```
10551
1835
3072
6150
1705
...
```

The usage for the program is:

```
A1 [-l] [-s] filename
```

where:

- `-l` is an optional command line argument that indicates that the output should be a letter count histogram. If it is not provided, display the simple number counts.
- `-s` is an optional command line argument that indicates that the histogram must be saved to the file, in addition to being displayed on screen. The saved format must follow the description above. It must be saved to the file `out.txt`. If the `-s` is not provided, do not save the output to the file.
- `filename` is the name of the file you want to analyze

For example:

- `A1 wonderland.txt` will display the simple number counts for the file wonderland.txt
- `A1 -l wonderland.txt` will display the letter count histogram for the file wonderland.txt
- `A1 -l -s wonderland.txt` will display the letter count histogram for the file `wonderland.txt` and save the letter count histogram to `out.txt`

Your code will be executed with mpiexec, e.g.

```
mpiexec -n 5 ./A1 -l -s wonderland.txt
```

Needless to say, the 5 nodes above are just an example. Your code cannot assume that it will be run on a specific number of MPI nodes - we may provide whatever number we want when we run `mpiexec`.

Two sample text files are provided: wonderland.txt (the full text for "Alice in Wonderland"[1],) and looking_glass.txt (the full text for "Through the Looking Glass"[2]). Use these files to test the program.

Also, you should probably create some other smaller text files to test your program's correctness.

Grading

- MPI-based histogram computation with simple number counts: **80%**.
  Note that, for each input file, we will try running your code with different number of MPI nodes, up to the max number of MPI nodes supported on our machines. Your code may also be tested with large multi-megabyte text files.
- Normalized letter count histogram: **10%**
- Error handling for MPI code, command line arguments, and files: **10%**

If your program does not compile, the assignment receives an automatic grade of **zero** (0). Up to **10%** of the marks will be deducted for any warnings displayed by the compiler.

Submission and Evaluation

Submit the assignment using Moodle. Submit only the source code and the makefile. Bundle the code in a zip file.

The assignment will be marked using the standard CIS*3090 Docker image provided on the course website and discussed in class. We will download and run a fresh image, create container, and use Docker to run and grade your code. Make sure you test your code accordingly.

Academic Misconduct
This assignment is individual work and is subject to the University Academic Misconduct Policy. See course outline for details.

[1] Obtained from Project Guttenberg: https://www.gutenberg.org/files/11/11-0.txt

[2] Obtained from Project Guttenberg: http://www.gutenberg.org/files/12/12-0.txt