Parallel Computing
Lab Assignment 1

In this lab you will write MPI code to create a few processes that will cooperate to generate a histogram. There are two inputs:

- The number of data points. This is a positive integer bigger than one and less or equal to one billion.
- The number of bins, which is a positive integer.

There will be no output generated, except the execution time taken, but there is a function that checks the correctness of the histogram, as you will see later in this lab description.

Example:

If the input is 5 and 3, the program will generate five floating point random numbers between 0 and 20, and three bins. This means the range of each bin = 20/3 = 6.7.
So, bin[0] will have the range [0, 6.7[, bin[1] the range [6.7, 13.4[, and bin[2] = [13.4, 20[

Let's assume that the data points generated are as follows:
2.5, 7.2, 16.8, 9.3, 18.1
The generated histogram will be:

histogram[0] = 1 histogram[1] = 2 histogram[2] = 2

To make your life easier, we have provided you with a file: lab1.c that does the following:

- Read the arguments from the command line (i.e. number of data items and number of bins).
- Dynamically allocate the arrays data[] and bins[]
- Fill out the data[] array with the random numbers from [0, 20[.
- Initialize array bins[] to 0
- Insert the code that will time your MPI code (do not change anything in that part).
- Check that your output histogram is correct by comparing it to a sequential version and print mismatched elements on the screen.

You will insert the code within lab1.c
Look at the place where there is a comment that starts with: TODO.

How will you parallelize this?

Suppose you have 100 data items, five bins, and four processes. This means that each process will be responsible for 100/4 = 25 data items. Process 0 will take care of elements data[0] to data[24], process 1 will take care of data[25] to data [49], etc.

Each process will form its own local histogram for the data it has.

Then you combine all these local histograms into one final histogram. This final histogram is the one that will be checked for correctness by process 0.

The report

After you implement your code, do the following experiments:

1. Draw a table as follows assuming the number of bins = 5. (Table entries represent the time.)

Processes →	1	2	4	8
# data items = 1 million				
#data items = 10 millions				

2. Draw a similar table but that contains the speedup relative to one process (i.e., time of one process / time of more than one process)
3. Draw a third table that contains the efficiency (speedup/#processes).
4. Answer the following questions based on the three tables above:
    a. What can you say about speedup patterns as you go from 1 to 8 processes for both 1 million and 10 million data items? Do you see a difference in the pattern when the problem size is bigger?
    b. Justify the pattern you explained in question "a" above.
    c. What can you say about the efficiency, in the third table, as you go from 1 million to 10 million? Do you see a difference in the pattern when the problem size is bigger?
    d. Justify the pattern you explained in question "c" above.

5. Draw a table as follows assume data items = 1 million: (Table entries represent the time.)

Processes →	1	2	4	8
# bins = 4				
#bins = 10				

7. What is the effect of increasing the number of bins on the performance? Justify.

Submission

- The report must be in pdf.
- The source code is in lab1.c
- Put the above two files in a zip file named netID.zip where netID is your own netID.
- Submit, as we do with all assignments, through Brightspace.

Enjoy!