Assignment (Maximum: 100 marks)
Due date: 10-04-2025

Problem: Time series data (values of a variable in spatial and temporal domain) of a 3D volume (nx * ny * nz) will be provided. The assignment is to find in parallel the (1) count of local minima (2) count of local maxima (3) global minimum and (4) global maximum for every time step. You may select any strategy to parallelize your code, keep performance in mind. You may perform any 3D (3 spatial dimensions). Please refer to lecture notes for 3D decomposition. An example of 3D domain decomposition along all 3 axes is shown. X-axis: Horizontal, Y-axis: Vertical, Z-axis: Going into the screen.

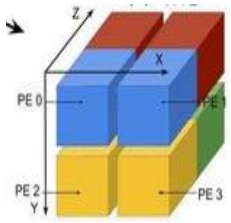

Figure source: https://www.researchgate.net/
Please ignore the annotations given on the figure.

Input file description: Input file will contain the time series data (floats) of a 3D volume. The number of rows represent the total number of grid points in the 3D volume (nx * ny * nz). The 3D volume may not be a cube. The maximum total number of grid points = 1024^3, the maximum number of grid points in any dimension is 1024. The number of columns represent the number of time steps in the data (maximum = 1000). Data will be organized in XYZ dimension order in the input file. An example is provided next. Let nx = 2, ny = 3, nz = 2, and number of time steps = 2, then the values in the file will be in XYZ order: (0,0,0), (1,0,0), (0,1,0), (1,1,0), (0,2,0), (1,2,0), (0,0,1), (1,0,1), (0,1,1), (1,1,1), (0,2,1), (1,2,1). The three tuples here show the (x,y,z) co-ordinates of the points. The input file will contain the corresponding data values (floats) and will look something like below:

```
aaa  bb
aa  bb
cbc  bcc
baa  bbb
abc  bc
ab  bcb
aaa  bbb
bc  abc
ab  bcb
baa  bbb
abc  bc
ab  bcb
```

Logical code sequence:

1. You may use 1 process (rank 0) to read the entire data from the input file using sequential input/output (I/O) function calls (for e.g., open/fopen, read/fread/fscanf/…, close/fclose etc.).
2. Rank 0 may distribute the data to all the processes, you may select any data distribution strategy.
3. You may select any process decomposition strategy to assign the sub-domains.
4. Every process finds the local minima and local maxima for its sub-volume. This may require communication with neighbouring processes.
5. The runtime input arguments are described next. Do not hardcode any input parameter in the code.

- Init

Time 1

- File read and data distribution

Time 2

- Main code

Time 3

- Output

Time 4

- Finalize

Timing your code: You must time the file read and data distribution time separately (steps 1 and 2). This is Time 2 – Time 1. For the main code, start timer (MPI_Wtime) after reading the file and stop timer before reporting the output. This is Time 3 – Time 2. Also report the total time (Time 4 – Time 1). Always, report the maximum time across all processes.

Input (nine arguments):

- Dataset (Input file name .txt)
- PX – Number of processes in X-dimension
- PY – Number of processes in Y-dimension
- PZ – Number of processes in Z-dimension
- NX – Number of grid points in X-dimension
- NY – Number of grid points in Y-dimension
- NZ – Number of grid points in Z-dimension
- NC – Number of columns (i.e. number of time steps)
- Output file name [preferably: output_NX_NY_NZ_NC.txt] [Note: *write output to this file from rank 0*]

- Note that P = PX * PY * PZ
  Note that number of processes P (specified using -n or –np depending on the system) is known while execution.

Constraints on input*:*

- *PX>=1, PY>=1, PZ>=1*
- *NX<=1024, NY<=1024, NZ<=1024*
- *NC<=1000*

Output: The output file should contain three lines (no extra lines to be printed). The number of 2-tuples in lines 1 and 2 depends on the number of time steps.

Line 1: (count of local minima, count of local maxima), ... // a pair of integer values per time step
Line 2: (global minimum, global maximum), ... // a pair of float values per time step
Line 3: Read Time, Main code Time, Total time // three doubles (across all time steps)

Preliminary execution instructions:

Run your code for the two example test cases provided (run 2 times each test case):

- mpirun -np 8 -f hostfile ./executable data_64_64_64_3.txt 2 2 2 3 output_64_64_64_3.txt
- mpirun -np 8 -f hostfile ./executable data_64_64_96_7.txt 2 2 2 7 output_64_64_96_7.txt

The entire code must be written using C+MPI. Name your source code as 'src.c'. Your code must be documented well. You must include a 'report.pdf' detailing the data distribution strategy and parallelization strategy you used with reasons. You must also describe your code in the report. There must be only 1 submission per group with full details of the group members and group name at the top of the report. A tex file template will be provided. More details regarding the report format, execution and submission will be provided.

Please refer to Lecture 1 regarding plagiarism policy.