LAB 2: MPI (C++)

Design due **2/11/16 - 11:59 pm**
Implementation due **2/19/16 - 1:00 pm**

In this lab you will implement the search for the longest **Palindrome** in a text file using MPI *and* implement the MPI_Gather function using only MPI_Send and MPI_Recv. Your program will use command line input to specify the number of nodes used. An MPI-how-to file will be provided describing details of compilation and execution.

A palindrome is word that is spelled the same both forward and backward; **ignore the case of the letters.** A palindrome doesn't cross from one line to another, you are only searching for the longest palindrome in a given line.

AbBa	Yes
ABCBA	Yes
ABCABC	No

We have provided a skeleton of the solution that reads in the text, strips it of any non-letter characters and stores the result in a matrix (lines) of size [100000][15]. The array is fixed and will be the same for every input.
Implement an array search using multiple processes, with each process working on separate slice of the array. The search will be executed as in Part A of Lab1. The input array of strings is sliced into *numProcesses* chunks that are passed to (*numProcesses*-1) worker processes (one chunk must be processed by the Master process). Each process receives a single chunk and processes it. It is important to note that only the Master process should do any input/output; none of the worker processes should read a file or write to a file/stdout.
You must use the function ProduceOutput that is provided in the skeleton to write the result of your search, **you cannot add or remove anything from the output, doing so will make you lose points.**

Example Text	lineNumber	firstChar	length
PabbAD Quote 99998 more lines	0	1	4
pOlo Risetovotesirthepeoplearewaiting Aspireototolalase 99997 more lines	1	0	13

There are 4 parts to this Lab

Part A Design:

File Name: Design.pdf
In a PDF file called Design.pdf provide a pseudo code implementation of Part B and Part C. For both parts your code must be as clear as possible, indicate clearly what you are doing, use comments if necessary. In both cases explicitly explain what code is executed by the Master process and what code is executed by the Worker process.

Part B MPI_Palindrome_Search:

File Name: partB.cpp
Provide a complete code that completes the skeleton and looks for the longest palindrome as described above. MPI_Scatter can be used by tge master node to distribute data to all nodes. To aggregate your results in the Master process **you must use MPI_Gather.**

Here is a good tutorial about it: http://mpitutorial.com/tutorials/mpi-scatter-gather-and-allgather/

```
int MPI_Gather(const void *sendbuf, int sendcount, MPI_Datatype sendtype, void
*recvbuf, int recvcount, MPI_Datatype recvtype, int root, MPI_Comm comm)
```

Reference: `http://www.mpich.org/static/docs/v3.2/www3/MPI_Gather.html`

Part C Custom_Gather:

FileName: partC.cpp

Provide a complete code to implement a custom Gather operation with the same semantics as the MPI_Gather. Use your Custom_Gather function in your MPI palindrome search code.

Your Custom_Gather operation must use only point-to-point communication, with nodes arranged in a virtual ring network (https://en.wikipedia.org/wiki/Ring_network). In a ring network each node is connected to only two other nodes, the node before it in the ring and the node after it. For example, if there are *n* nodes then node 0 is connected to node 1 and node *n-1,* node 1 is connected to node 0 and node 2, node 2 to node 1 and node 3 and so on until node *n-1* is connected to node *n-2* and node 0. Thus the whole network forms a ring, with only the nearest-neighbor direct communication.

Part D Analysis:

File Name: analysis.pdf

Run your code of Part B and Part C on the files Timing_1, Timing_2, and Timing_3 (located in the EEE dropbox) and time how each performs using 2, 4, 8, and 16 processes. Be careful when timing your code, you want to time only the processing time, not reading the input or writing the output. Only the Master process should time the execution, not the worker threads. Graph them and explain what they show.

How does the input data affect the result?

How does the number of processes affect the result?

What is the topology used by MPI to implement its Reduce?

How does the ring topology affect the runtime?

How many messages are passed between nodes in each part?

Submit via EEE (dropbox "Lab1") **in 2 parts:** *YOU **MUST** USE the file names below*

1. Part A must be submitted as a PDF file called **Design.pdf** in Dropbox Lab 2 - Design
2. Implementation of your C++ program for parts B and C in files **partB.cpp** and **partC.cpp** respectively.
3. Analysis must be submitted as a PDF file called **Analysis.pdf**

Point Breakdown:

Part B: Design 10pts, Implementation 30pts

Part C: Design 10pts, Implementation 30pts

Part D: Analysis 20pts.

Example input Files to use in your Implementation will be uploaded to the sub-folder "Course Files" in the Dropbox Folder "Lab 1 - Samples".

NOTE: The TA cannot solve the exercises for you, but he can answer your questions!