CSCI 176 (Parallel Processing) Spring 2021 Prog. Assignment 5 (15 pts) Due: April 29 (Th)

Parallel merge sort with MPI.

Do the programming assignment #3.8 in pp. 148 of the basic textbook, i.e.,

"Parallel merge sort starts with n/comm_sz keys assigned to each process. It ends with all the keys stored on process 0 in sorted order. To achieve this, it uses the same tree-structured communication that we used to implement a global sum. However, when a process receives another process's keys, it merges the new keys into its already sorted list of keys. Write a program that implements parallel merge sort. Process 0 should read in n and broadcast it to the other processes. Each process should use a random number generator to create a local list of n/comm_sz ints. Each process should then sort its local list, and process 0 should gather and print the local lists. Then the processes should use tree-structured communication to merge the global list onto process 0, which prints the result."

Programming guide:

1. Process_0 accesses command line argument for n (total number of elements in the list), and broadcast it to all other processes;
2. Each process creates a dynamic array of integers with n/p elements, and fills it with random integer numbers; for the random numbers, please generate each number less than 100;
3. Process_0 collects the local lists (unsorted) from other processes and displays them in order by rank_id;
4. Each process performs qsort (GNU library quicksort) to sort the local list in the ascending order; please study any Internet sources for the usage of the qsort(..), if you haven't used it so far; Note: please do not use the ridiculous recursion based approach for sorting the local list;
5. Implement the tree-reduction for the merging operation, i.e., merging locally sorted lists into one; you should use the method developed for HW1(#3), i.e., finding partner and send or receive, etc.; receiving processes should perform the merging operation and yield double-length merged list in each step;
6. Finally, process_0 produces the global sorted list and displays it.

Submission:

Before compilation, include good documentation (global doc, each func head doc, etc.) in the source code; and submit source code and screenshot(s) of execution output.

For the output to submit, please use n=400 and test with p=2, 4, 8.
Sample output is shown below.

Sample output (n=40, p=4 case):

```
process_0: 83 86 77 15 93 35 86 92 49 21
process_1: 90 19 88 75 61 98 64 77 45 27
process_2: 46 85 68 40 25 40 72 76  1 64
process_3:  1 83 74 26 63 37 25 63 28 85

sorted list:
 1  1 15 19 21 25 25 26 27 28 35 37 40 40 45 46 49 61 63 63 64 64 68
72 74 75 76 77 77 83 83 85 85 86 86 88 90 92 93 98
```