Lab #4 MPI Programming III

Purpose: to learn how to use collective operations and write more complex MPI programs.

1. (35) write an MPI sorting programming.

- You need to write two programs:
  - seq.c : sort arrays locally (it includes bubbleSort or quicksort, see below)
  - para.c : parallel version with MPI using **odd-even sorting**
    1) Process 0 generates the whole array, distributes it to other processes including itself, collects the sorted subarrays, and **merges** them into one final sorted array with length n
    2) Use *collective* operations for array distribution and subarray collection
    3) After the whole array is distributed to different processes, all the processes should be involved when all the numbers are being sorted using odd-even sorting
    4) Use different number of processes to run the program

- Requirement:
  - The array size should be passed in from command line, such as mpirun -np $NPROCS -machinefile $PBS_NODEFILE $MCA_OPTS $PROGRAM ***ARRAY_SIZE***
  - Use random number generators to create unsorted arrays
  - Write a function called *void bubbleSort(int *array, int n)* to sort integer arrays/subarrays locally
  - If n <= 40, print out the sorted array. You can assume n can be perfectly divided by npes (number of processes).
  - Print out the execution time (NOT include array generation time) in process 0 (set timers before array distribution and after getting the final result), and fill out the following form:

	1 process	4 processes	8 processes	16 processes
40000 elements				
400000 elements				

  - Change your local sorting algorithm to *quicksort* and fill out the following form:

	1 process	4 processes	8 processes	16 processes
400000 elements				
4000000 elements				

  - Analyze and explain the data in the two tables.