General Submission Guidelines

Files. C or C++ code files, and a PDF report file should be submitted on Gradescope.

Submission Code Structure. Make a different sub-directory for each question with the name question#. Every question sub-directory should be independently buildable. You can re-use files through symlinks or relative paths in your Makefile. A simple flat structure is preferred. For example,

```
hw3
|-- question1
    |-- main.cpp
    +-- Makefile
+-- question2
    |-- main.cpp
    +-- Makefile
```

Development and Testing. You can develop and test your code on your local machines with gcc. However, before submitting your homework, make sure your programs compile and run on mc19.cs.purdue.edu.

Grading. mc19 machine will be used for testing your code with GNU compiler. Output your results to `stdout` using `printf()` or `cout`. If your code needs a different build process other than the one we will use as default, you will have to include a Makefile in the question sub-directory with the default target building the executable.

Extensions. No late submissions will be accepted, unless students have prior extensions from the instructors.

Homework Specific Guidelines

Build Command. By default, we will compile all of your C/ C++ files in each question sub-directory with the command `mpic++ *.cpp -o ./run`. All of your source files in the question's sub-directory will be built into an executable that outputs to `stdout`. If you do not want to include a faulty source file for grading, remove it from the sub-directory; otherwise it will fail the default build command.

If you do not want to us to use the default build command, include a Makefile in each question's sub-directory. However, it is recommended to use the simplest flat-file structure for your code as shown in the general submission guidelines and thus focus on experiments and results, not software engineering. You will still need to create sub-directories for questions.

Tools. You cannot use any language other than C/ C++ for implementing the algorithms. Data analytics/ plotting can be done with other tools. Excel or Google Sheets are the easiest to use for plotting graphs.

Execution. If your program uses command line arguments or some sort of input, you need to explain how to use your program in your report PDF. It is preferable if you avoid command line arguments, or automate the inputs to these arguments in a Makefile target or shell script included with your submission (e.g. you can include a shell script called `run.sh` that runs the program with different command line arguments).

Output. We will execute your programs on mc19. Make sure your code compiles, runs, and does not throw errors like segfaults on the test environment.

Q1. (20 points) Ping pong messages of increasing lengths ($m = 1K \ldots 100K$). Compute the round-trip time (RTT). Divide by two and the number of ping pongs to compute communication time. Plot these times on a line to estimate startup time t_s and per-word transfer time t_w.

Q2. (40 points) Repeat the experiment, but in this case, each processor is paired with another processor (you can do this simply by pairing the low half of the processors with the high half). All processor pairs now ping pong messages. Estimate t_s and t_w for different values of p $(2, 4, 8, 16, 32)$. Plot the dependence of t_w on p in this congested case.

Q3. (40 points) Implement the sorting algorithm on slide 60 of the slides on sorting (message passing quicksort). Plot the speedup as a function of list size (i.e., fix p at different values (2, 4, 8, 16, 32) and the list size on the X-axis. Also plot speedup as a function of p (i.e., fix list sizes at different values – $100K, 1000K, 10000K$ and plot p on the X-axis).

Guides To compile an OpenMPI program, you can use `mpicc` or `mpic++` instead of `gcc`. For executing your programs, use `mpirun` with `-np <int>` flag to specify the number of processes. Refer to this tutorial for examples. You can find API documentation here.

Machine usage Here, since we ask you to go up to $p = 32$, please use mc19 for all of your testing as this particular server has enough available slots to accommodate the number of processes requested.