Ankai Lou
CSE 5441 — Parallel
Instructor: J. Jones

Lab # 4: MPI

To compile on the OSC cluster:

```
> mpicc -O lab4.c
```

To run with 2 processes on the OSC cluster:

```
> mpiexec -np 2 a.out
```

The runtime (calculated using `clock()` or CPU-time) and bandwidth for the message sizes of `i = 32, 256, 512, 1024, 2048` double-precision floating-point numbers:

Message Size	Run Time	Bandwidth
32	0.762500 seconds	335.737701
256	1.147500 seconds	1784.749390
512	1.620000 seconds	2528.395020
1024	2.620000 seconds	3126.717773
2048	5.372500 seconds	3049.604492

The runtime (measured in CPU-time to ignore blocking time for `MPI_Send` and `MPI_Recv`) increases and message size increases — this is intuitive as the larger the buffer, the longer the time to ping-pong the buffer back and forth 1000000 iterations. For smaller message sizes — i.e. the shift from 32 to 256 — the runtime does not seem to scale linearly with respect to message size, rather logarithmically. For larger message sizes the runtime seems to scale linearly. This means that increasing the buffer size is beneficial for smaller message sizes — but reaches a point of diminishing returns as message size increases.

The bandwidth seems to increase with respect to runtime up until the shift from 1024 and 2048 — where the bandwidth drops. This suggest that the point of diminishing returns for buffer-size occurs somewhere between 1024 and 2048 double-precision floating-point numbers. Therefore, the ideal size for an MPI buffer for ping-ponging messages between two processes seems to be around `sizeof(double) * 1024` bytes to `sizeof(double) * 2048` bytes. This is consistent with the runtime behavior going from logarithmic to linear near the same point in the trials.