Project 2

Due: October 20 (Sunday), 2024 at 11:59 pm.

Please read the Common Projects Instructions before doing any problems.

Problem 2.1

The MPI implementation several dozens of collective operation functions such as broadcast, gather, scatter, all-gather, etc. Describe the algorithms of implementation five functions: MPI_Bcast(), MPI_Scatter(), and MPI_Allgather(), MP_Alltoall() and MPI_Reduce().

MPI_Reduce() has the option, among many others, to locate the global min. This project requires you, using everything but the MPI_Reduce(), to compose MY_Global_Min_Loc() for this purpose, for a computer with P cores each having a local array of length N. Your answer will be an array of length N containing processor ranks as the location of min's. For example, for a parallel computer with $P = 3$ processors and each with an array of $N = 4$ elements, we may have the following:

For Proc $P = 0$, $A_0 = \{1, 9, 3, 4\}$
For Proc $P = 1$, $A_1 = \{5, 6, 7, 2\}$
For Proc $P = 2$, $A_2 = \{9, 8, 6, 1\}$

For the $A_0[0], A_1[0], A_2[0]$ elements, the min (1) resides on Proc 0.
For the $A_0[1], A_1[1], A_2[1]$ elements, the min (2) resides on Proc 1.
For the $A_0[2], A_1[2], A_2[2]$ elements, the min (3) resides on Proc 0.
For the $A_0[3], A_1[3], A_2[3]$ elements, the min (1) resides on Proc 2.

Thus, you need to report an array of ranks $\{0, 1, 0, 2\}$.

Please be advised that we are NOT requiring you to compute the min within each process, e.g., for Proc $P = 0$, $A_0 = \{1, 9, 3, 4\}$*, the min is "1". We knew it at the PPP (pre-parallel processing) dynasty.*

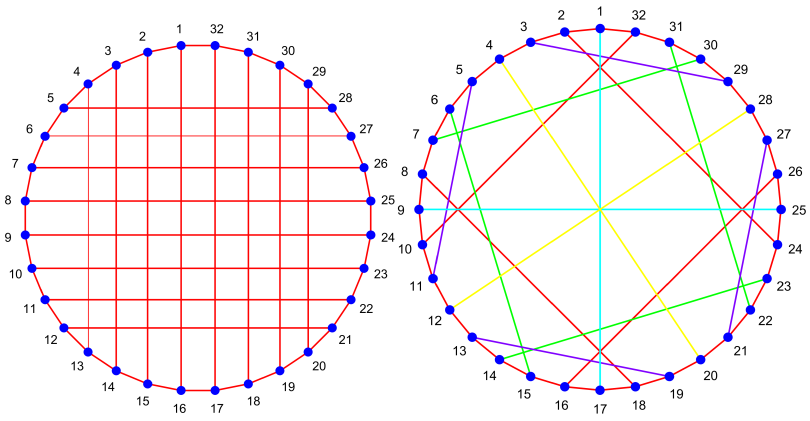
Problem 2.2

In the following two graphs, named as Bid32 (left) and Opt32 (right), the balls are considered as identical vertices (i.e., computer cores) and lines (regardless of colors or thickness) are considered as directionless edges (i.e., computer network links over which data traverse bi-directionally). Many edges appear overlapping, but they are not connected; all edges must start and end at vertices. In the Opt32, the edges are of various colors, and they are for an irrelevant purpose.

Please design parallel algorithm(s) to send $N = 10$ floating-point numbers (e.g., 1.1314, 3.1415, 250.250…) from each vertex (node) to all other vertices for both "topologies", i.e., you initiate your 32 nodes with N random numbers, different on different nodes. After your all-gather operations, every node will have $32 \times N = 32 \times 10$ numbers (including own).

Construct a table to show the time requires to complete such all-gather operations:

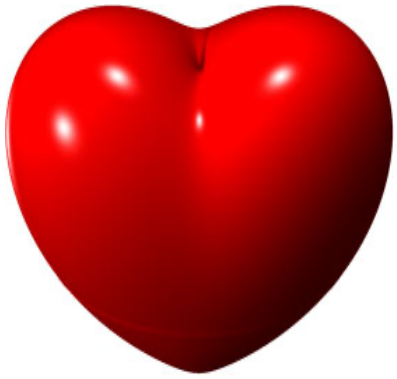
	On Bid32	On Opt32
$N = 10$		
$N = 20$		
$N = 30$		
$N = 40$		



Problem 2.3

I made you a heart expressed in equation,

$$\left(x^2+\left(\frac{3}{2}y\right)^2+z^2-1\right)^3-\left(x^2+\frac{1}{50}\left(\frac{3}{2}y\right)^2\right)z^3=0$$



Obviously, what's given is the 3D surface equation of the heart and the assumption that the heart has no "cavities" and, thus, its density is uniform, etc. However, the heart pumps blood in and out and its volume, surface area, mass (including blood in it) is expected to vary with time. With such in mind, "crazy" problems can be hacked to make this problem "more" fun and harder, at least, for parallel computing.

1. Design a parallel algorithm/program to compute the surface area of the heart with the given surface eq.
2. Design a parallel algorithm/program to compute the heart's mass at time $t=\pi/6$ if the heart mass density is,

$$\rho(x,y,z;t)=\rho_0(x,y,z)\left(\frac{1+\sin\omega t}{2}\right)$$

where ω is the heart rate. Of course, I made up the formula and still do not how $\rho_0(x,y,z)$ looks like. You may create a density function that may make some sense. The trivial case (required for our project) is $\rho_0(x,y,z)=1$.

You perform minimum necessary calculations, using $P=1,4,16$ cores, to achieve 3 digits of accuracy. Report the number of floating-point operations per core (you may average) for each case.

Results (floating-point operations per core) might look like,

To compute:	P=1	P=4	P=16
Surface area			
Heart mass			