3.0

Implement matrix-vector multiplication using MPI. You can have process 0 read the matrix $\boldsymbol{A}$ and the vector x. Assume the matrix is square of order n and that n is evenly divisible by comm_sz.

> **INPUT**: n – the dimension for $\boldsymbol{A}$ and x.
>
> **(**You can use a n^2 vector to denote the $n \times n$ matrix)
>
> **OUTPUT**: The result vector $y = \boldsymbol{A}x$.
>
> If you use program-generated random numbers for $\boldsymbol{A}$ and x, please also output $\boldsymbol{A}$ and x, and $\boldsymbol{A}$ in matrix format.

3.1

Suppose we toss darts randomly at a square dartboard, whose bullseye is at the origin, and whose sides are 2 feet in length. Suppose also that there's a circle inscribed in the square dartboard. The radius of the circle is 1 foot, and it's area is π square feet. If the points that are hit by the darts are uniformly distributed (and we always hit the square), then the number of darts that hit inside the circle should approximately satisfy the equation

$$\frac{\text{number in circle}}{\text{total number of tosses}} = \frac{\pi}{4},$$

since the ratio of the area of the circle to the area of the square is $\pi/4$.

We can use this formula to estimate the value of π with a random number generator:

```
number_in_circle = 0;
for (toss = 0; toss < number_of_tosses; toss++) {
   x = random double between −1 and 1;
   y = random double between −1 and 1;
   distance_squared = x*x + y*y;
   if (distance_squared <= 1) number_in_circle++;
}
pi_estimate = 4*number_in_circle/((double) number_of_tosses);
```

Use OpenMPI to write a parallel program for this method. The processes evenly distribute the tosses, while process 0 sums up the total number of darts which hit inside the circle from processes, calculates and outputs the estimate of the value of π.

INPUT: n – the number of total tosses.

OUTPUT: The estimate of the value of π.