5.1.

Modify the matrix-vector multiplication so that each thread uses private storage for its part of `y` during the `for i` loop. When a thread is done computing its part of `y`, it should copy its private storage into the shared variable.

5.2.

Write a Pthreads program that implements the histogram program in Chapter 2.

5.3.

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);
```

This is called a "Monte Carlo" method, since it uses randomness (the dart tosses).

Write a Pthreads program that uses a Monte Carlo method to estimate π. The main thread should read in the total number of tosses and print the estimate. You may want to use `long long ints` for the number of hits in the circle and the number of tosses, since both may have to be very large to get a reasonable estimate of π.