Name: Nikhil Kumar Gattu
Mail: gattun@oregonstate.edu
ONID: 934615235

1. **Tell what machine you ran this on**

Ans: I ran the experiment on flip4.engr.oregonstate.edu server, which supports OpenMP version 201511.

2. **What performance results did you get?**

Ans:

```
...4:~ — ssh gattun@access.engr.oregonstate.edu
flip4 ~ 78$ bash loop.bash
OpenMP version 201511 is supported here
For 1 threads, Peak Performance =   160.92 MegaMults/Sec
OpenMP version 201511 is supported here
For 4 threads, Peak Performance =   615.54 MegaMults/Sec
flip4 ~ 79$
```

1 thread: 160.92 MegaMults/Sec
4 thread: 615.54 MegaMults/Sec

3. **What was your 1-thread-to-4-thread speedup?**

Ans: S = 615.54/160.92 = **3.82**

4. **Your 1-thread-to-4-thread speedup should be less than 4.0. Why do you think it is this way?**

Ans: I think real-world performance is always affected by:

- Overhead in thread management (creating and scheduling threads)
- Memory bandwidth limitations
- CPU cache sharing
- Small portions of the program that remain sequential (as explained by Amdahl's Law)

These factors prevent us from reaching perfect linear scaling.

5. **What was your Parallel Fraction, Fp? (Hint: it should be less than 1.0, but not much less.)**

Ans: Fp = 4/3 x (1 – 1/3.82)

= 4/3 x (1 – 0.262)

= 4/3 x 0.738

= **0.984**

This means about 98.4% of the code is parallelizable, which is excellent and expected since the array multiplication is a highly parallel task.