Nicholas Grout
HW4
CS201

1. **What version of the code provided the best performance? (non-optimized, unrolled 4 times, or unrolled 8 times)**

The code with 4 loop unrolls usually had the best performance. 8 loop unrolls was slowest on every machine, and no loop unrolls was usually slower than 4 unrolls, but was sometimes just as fast as 4 unrolls. On my computer (ubuntu 16 with i7-4790k processor) 4 unrolls performed the same as no unrolls. However on The CAT's "ada" server (address: linux.cs.pdx.edu) the results were much more distinct and consistent, with 4 loop unrolls being fastest, no loop unrolls being second fastest, and 8 loop unrolls slowest.

2. **Did loop unrolling provided any benefit?**

Yes, 4 loop unrolls overall performed faster than no loop unrolls. However 8 loop unrolls performed slower than no loop unrolls. This might be because of the additional overhead of loop unrolling, in combination with the limitations of using 8 loop unrolls. When loop unrolling, we must compute the remainder of our matrix's dimension divided by our loop unroll number. When we do this with 4 loop unrolls and a 12 by 12 matrix, we can do all of our computing in the loop-unrolled loop, leading to faster performance. However when we have longer loop unrolling, we still have the overhead but we can't use our unrolled loop as long and must use the regular loop.

For our matrices, we had a dimension of 12. This means that we could only use our 8 unrolls loop for ⅔ of the equations. However if we unroll 4 times, we need to do more loops but we can use the unrolled loop for all computations. This is important when thinking of things like computer graphics, which frequently uses large fixed-size multidimensional arrays. This means that calculations for a computer display could be optimized through loop unrolling just like our 12x12 matrix, only it might be an array of size 1920x1080.

3. **Why do you think that loop unrolling was or was not beneficial in this case?**

4 loop unrolls was clearly beneficial in this case. We saw a significant and consistent performance increase when we unrolled 4 times. It would be interesting to see how different matrices dimensions affect this however. At what matrix dimension does 8 loop unrolls become more efficient than 4 loop unrolls? Would this ever happen?

I use loop unrolling in my own programming as well. I recently implemented a b tree for part of a pet project, and saw significant performance increases in finding data. Since each node in this case has less few data points (2-5ish) it's actually faster to use an accumulator and compare all data items in this node than to use an if statement and stop once we find a match.