EfficeintML.ai Lab 5 Report
Name:_____

1. Loop Unrolling (20pt): Please fill in the starter code in *kernel/template/loop_unrolling.cc* to implement loop unrolling and run the `./evaluate.sh loop_unrolling` to evaluate performance improvement.
    a. Please copy and paste your implementation in *kernel/template/loop_unrolling.cc*: (15pt)

    b. How does the performance in GOPs, achieved through loop unrolling on your computer, compare to the reference implementation? Please explain the performance difference. (5pt)

2. Multithreading (20pt): Please fill in the starter code in *kernel/template/multithreading.cc* to implement multithreading and run the `./evaluate.sh multithreading` to evaluate performance improvement.
    a. Please copy and paste your implementation in *kernel/template/multithreading.cc*: (15pt)

3. How does the performance in GOPs, achieved through multithreading on your computer, compare to the reference implementation? Please explain the performance difference. (5pt)

4. SIMD Programming (20pt): Please fill in the starter code in *kernel/template/simd_programming.cc* to implement SIMD programming and run the `./evaluate.sh simd_programming` to evaluate the performance improvement.
    a. Please copy and paste your implementation in *kernel/template/simd_programming.cc*: (15pt)

b. How does the performance in GOPs, achieved through SIMD programming on your computer, compare to the reference implementation? Please explain the performance difference. (5pt)

5. Multithreading with Loop Unrolling (20pt): Please fill in the starter code in *kernel/template/multithreading_loop_unrolling.cc* to implement multithreading and loop unrolling and run the `./evaluate.sh multithreading_loop_unrolling` to evaluate the performance improvement.
    a. Please copy and paste your implementation in *kernel/template/multithreading_loop_unrolling.cc*: (15pt)

    b. How does the performance in GOPs, achieved through multithreading and loop unrolling on your computer, compare to the reference implementation? Please explain the performance difference. (5pt)

6. Combination of All Techniques (20pt): Please fill in the starter code in *kernel/template/all_techniques.cc* to implement all techniques above and run the `./evaluate.sh all_techniques` to evaluate the performance improvement.
    a. Please copy and paste your implementation in *kernel/template/all_techniques.cc*: (15pt)

    b. How does the performance in GOPs, achieved through all optimization techniques on your computer, compare to the reference implementation? Please explain the performance difference. (5pt)

7. Bonus (20pt): Any optimization techniques on your mind? Try to implement them to improve the performance further! If you can further improve the performance compared to the optimized kernel in TinyChatEngine, you can get bonus points here! Each percent of performance speedup equals one point (create a pull request in the repo and get verified by the TA), up to 20 points.