Question 3: Optimizing Cache Usage in Scientific and Machine Learning Workloads

Large-scale computations in scientific applications (e.g., matrix multiplication, weather simulation, training neural networks) often face **cache bottlenecks** due to their use of large data arrays and regular memory access patterns. Your task is to investigate how modern systems (CPUs, GPUs, or edge accelerators) can improve memory locality and cache usage to speed up such workloads.

Choose a representative workload such as:

- Matrix multiplication in a BLAS library (e.g., OpenBLAS, Intel MKL)
- CNN inference or training (e.g., in TensorFlow, PyTorch)
- Data-Parallel scientific computing (e.g., FEM, CFD, or simulation kernels)

In your report:

- Explain why cache bottlenecks arise in these workloads (e.g., large working sets, strided memory access)
- Propose software-level optimization strategies such as:
  - Loop tiling/cache blocking
  - Prefetching (manual or compiler-inserted)
  - Loop interchange or memory layout changes
- Optionally, evaluate compiler-assisted approaches (e.g., polyhedral models) or accelerator-specific tactics (e.g., tensor cores, local shared memory)
- Use a real example or pseudocode to show how your strategy improves spatial/temporal locality
- Include expected impact on cache hit ratio, latency, or performance improvement (e.g., "blocking matrix multiplication improves L1 cache hit rate from 60% to 90%)
- Finally, reflect briefly on how better cache utilization contributes to energy savings or longer hardware lifespan, especially in constrained environments like edge devices

Expected Outcome:

- Identify sources of memory inefficiency in high-computation workloads
- Apply textbook concepts like locality, blocking, associativity, and reuse distance
- Propose cache-aware code transformations
- Estimate impact on real systems (e.g., reduced DRAM access, lower latency)
- Relate memory efficiency to energy efficiency or hardware scaling

Suggested Readings:

- Patterson & Hennessy – Computer Organization and Design: RISC-V Edition [Chapter 5: Large and Fast - Exploiting Memory Hierarchy]
- https://www.intel.com/content/www/us/en/developer/overview.html