Data Structure:

Similar to the 2nd assignment (using an array of Trie nodes to emulate a linked list), I utilized an array of Queues. Unlike Stacks, the Queue data structure is open on both ends: the rear, which is the location in which data is inserted (enqueue) and the front, which is the location in which data is removed (dequeue). This aspect of this unique data structure allows for an easier utilization of the First-In-First-Out (FIFO) methodology, since items are stored in the order they are inserted and can be deleted by that specific ordering (the front, which is where items are deleted from, stores the items that were inserted first while the rear stores the items that were recently inserted into the Queue).

Observations (Cache A & Cache B):

By analyzing the outputs in the provided test cases, it is clear that Cache B, which utilizes an arrangement of [index, tag, block], performs significantly more miss operations than Cache A, which utilizes an arrangement of [tag, index, block]. When handling small test cases, such as trace file 1, Cache B displays fewer hits and slightly higher misses, ranging from the tens to the hundreds, than Cache A. However, when test cases become extremely larger, such as trace file 3 and 4, Cache B begins to display significantly lower hits and higher misses, which range from a few thousand to several hundred-thousand or even higher with test cases that consist of millions of data. By analyzing, calculating, and comparing these results/outputs, we can see that the cache-hit ratio and cache-miss ratio of Cache A, which utilizes write-through, FIFO, and the arrangement [tag, index, block], are both approximately 2/3 while the cache-hit ratio and cache-miss ratio of Cache B, which utilizes write-through, FIFO, and the arrangement [index, tag, block], are 1/3 and 2/3 respectively.