Kyan Kotschevar-Smead
011805147
PA 1 CPTS 360 Cache Simulator

Implementation Details:

As part of the implementation I have 3 components. The first component of the program is gathering the command line arguments. In my cacheio.* I define functions and a struct for gathering the command line arguments. I then initialize my cache with command line args, where I use the arguments to dynamically allocate space for cache sets and lines. I do the same with my CacheStats struct to keep track of stats (hits misses evictions). To run the simulator I implement an iterator pattern that obtains one line at a time, a memory access struct that I pass use in my simulation calls. This is contained in a while loop and thus, as long as there are memory accesses the simulation will keep running. This pattern was important for not having to allocate memory and not knowing the size of the file. With everything set up I then make calls on the cache with the preformAccess function. Since read and write are the same in the simulation, and there are no instruction loads, preformAccess calls a function read_write on which checks for a hit, miss, or miss/eviction. And since modify is just read_write twice, preformAccess just calls read_write twice on modify. Depending on the result of the calls to read_write, preformAccess will update stats accordingly. This procedure is repeated until the whole trace file is read and thus conducting the simulation. For more specific details on function implementation please see code comments