Our group chose to implement Simulation Exercise 2: Main Memory Allocation.

In order to make our version of this simulation, we chose to implement a single linked list format with 128 nodes, each representing a 2KB block, for a total of 256KB. 10,000 processes are generated and are then randomly chosen to request a number of nodes for their own use (minimum of 3, maximum of 10 per process). This process is repeated for each memory management algorithm (10,000 requests each). (First Fit, Best Fit, Next Fit, Worst Fit.) During each of these processes, statistics for each algorithm are collected such as total memory fragments (holes of size 1 or 2), Average Fragmentation (total number of fragments / total allocations), average node allocation time, number of allocations denied, percentage of allocation requests denied, and number of deallocations. Once the processes are completed, the memory driver collects and outputs all the statistics and outputs them to the console for each individual algorithm. Our memory component includes 3 different types of methods, however, the only method that differs between each of the algorithms is the allocation method, which gets an override in each of the memory subclasses, while the deallocation and fragment-counting methods are implemented in the same way for each of the memory subclasses, which are inherited from the Memory superclass.

This project was both challenging and fun. It gave us an opportunity to use our skills and knowledge to implement a complex memory management simulation algorithm. We were very happy to be writing code again.