Robert Bonagura and Peter Tilton
20 October 2019
CS 214: Systems Programming
Assignment 1

Design of myblock and metadata:

- Our metadata is 2 bytes long, in the form of a signed short. The magnitude of the short describes the size of the associated block, and the sign denotes whether or not the block is in use. For example, a value of -256 means that the associated block has 256 unused bytes.
- The structure of the 4096-char memory array we use takes a repeating metadata-data approach. Each block of memory is immediately preceded by a metadata struct, whether or not it is in use. For example, after one block of size 60 is requested, myblock would look like: metadata 0 (2B) … data 0 (60B) … metadata 1 (2B) … metadata 1 (4032B).
- When mymalloc() is called with a valid user size, the first available block of sufficient size is split into two, with new metadata placed after the now-used block to indicate the updated size of the unused block. A pointer to the initial block is then returned to the user.
- When myfree() is called, adjacent unused blocks are stitched together into a single unused block until stitching can no longer be performed.

Workload data and findings:

- Average time for workload A: 0.008270 milliseconds. It makes sense for this to be the most quickly executed workload, as myblock is always empty when mymalloc() is called, so the first block will be available and no additional traversal will be necessary.
- Average time for workload B: 0.077410 milliseconds. This took longer than workload C and D, which makes sense considering traversal--adding one block at a time to memory, it will take O(n) time to add a block, and O(n^2) time overall. The later methods mix in free() calls, which make available blocks appear earlier in memory, requiring less traversal.
- Average time for workload C: 0.014770 milliseconds. This took longer than workload A because the user pointers needed to be stored and referenced from an array, adding memory access time.
- Average time for workload D: 0.020080 milliseconds. This took slightly longer than workload C, as the larger sizes of requested blocks means that myblock will eventually fill up, causing a more irregular memory structure that results in more stitching and traversal.
- Average time for workload E: 1.656650 milliseconds. This workload took the longest time, as it is designed to entirely fill up memory and then entirely free it. The total number of mymalloc() calls is therefore highest for this workload.
- Average time for workload F: 0.720140 milliseconds. This took between workload D and workload E to complete, which makes sense--this workload performs similar tasks to D, but with the additional weight of putting values inside the user blocks and printing them out.