Own Version of Malloc – Assignment 2
Scherer, Wood

The goal of this project was to mimick the malloc and free built in functions of C. We accomplish this by using a static char array as our “internal” memory in which acts as our space to be used. At the same time, we utilize structs called MemEntries which help us keep track of each malloc or free, otherwise it would be hard to do with just an array.

mymalloc: Depending on the size of the requested allocation, mymalloc will start at the very beginning of the memory block or the middle of the memory block using our two static MemEntry pointers which allow us to split up larger chunks of memory from smaller chunks, so that fragmentation doesn't happen. So memory less than 50 goes to the smaller block (first half), and anything larger than 50 goes to the bigger block (second half). Then we iterate MemEntries starting at the smallMemPtr or bigMemPtr until we find a free MemEntry with a large enough size. If found, we basically split that MemEntry so that the new allocation becomes the first entry, and the leftover memory from that original MemEntry becomes the next entry. If there isn't a free MemEntry or there is not enough space, then we cannot successfully allocate memory and return NULL.

Big Oh Complexity: mymalloc runs in O(n/2) ~ O(n) time. This is from our linear iteration across MemEntries where n is the total number of MemEntries. It is initially n/2 because we only iterate half of the memoryblock at most when we check if it is a large or small allocation. But we can call it O(n) as n becomes large.