Anthony Chou / **Vijay Appasamy**

Assign1 myMalloc/myFree

In this assignment , we implemented our own version of the library calls malloc() and free() that exist in C. We used a static char array with a fixed size of 5000 to represent the location where memory actually comes from when using malloc and free. We also made an array called NodeARR which contains the void pointers that point towards the blocks of memory, which is calculated by the max allocatable memory divided by the size of the struct MNodeLL that we made. All elements of the array are initialized to 0 which is basically null, and the reason for making them null is to keep track of the memory malloced by our mymalloc. This can also be used in detecting for errors like trying to free something that hasn't been malloced or at least malloced by our implementation. In our header we defined myfree and mymalloc as malloc and free so if mymalloc.c is included, it is possible to just use malloc() or free() like the library calls in order to use our implementation. When you malloc(), a chunk of space is set aside based on the inputted size, which is stored into our MNodeLL struct header which allows us to check for the amount of space used and whether or not the requested size of space is unallocated and free to malloc. The pointer to MNodeLL where the size and allocation flag is stored is stored in the first free index of the NodeARR array. When you use the free() call, the size of the struct is subtracted from the address contained within the pointer and is checked with the array for whether or not the pointer was actually malloced(the allocation flag). If the pointer was malloced, the free command frees up that block of memory and merges it together with the other free blocks and sets the allocation flag to 0. However, if the pointer has not been malloced, an error is returned, but if it was malloced (0) then a null is returned.