CS 214: Systems Programming, Fall 2020
Assignment 1: ++Malloc

In this assignment, we dive further into the usage of malloc() and free(), and how they aid in dynamic memory allocation. By implementing our own versions of these two built-in C functions, we can better understand the syntax of malloc, and understand how it returns a pointer value. The free() function allows the programmer to avoid memory leaks and have robust, non-faulty code.

mymalloc.c
This source file contains the primary functions of mymalloc() and myfree(). The helper function, firstFree(), searches through the myblock array for the first free memory space available. This is needed for adjusting the pointer location once mymalloc() or myfree() is called. Within mymalloc(), the program handles errors such as insufficient memory space, or allocation size being 0. If no errors are encountered, the function allocates the specified size (bytes) within the array memblock[4096]. Likewise, myfree() sets the address for malloced memory to be freed, and handles errors such as null pointer, un-malloc()ed space, or redundant myfree() calls on the same pointer.

mymalloc.h
Header file contains the macros as given in the Asst1.pdf file:
#define malloc (x) mymalloc(x, __FILE__, __LINE__)
#define free(x) myfree(x, __FILE__, __LINE__)
The metadata is contained within a struct of two ints (if an entry is free, and its size) and two struct pointers for the sake of creating a doubly linked list out of myblock[]. Using the sizeof keyword, we see that the entry data size may be either 2 or 4 bytes, depending on if the next/prev pointers are set.

memgrind.c
This file contains the workloads A–E, which test the mymalloc() and myfree()functions. The design choices behind workloads D and E are explained in testcases.txt. The output of this file prints the errors encountered during execution, the mean time elapsed for each workload, and the overall average time for all five.

Makefile
After consulting the GNU documentation, we created the makefile using @echo commands and used check flags to compile mymalloc.c and memgrind.c, make the corresponding object file mymalloc.o, and create the final executable, memgrind.