Project Title –Assignment 1: ++Malloc
Class: CS 214: Systems Programming, Fall 2020

Introduction

In this assignment, you will implement malloc() and free() library calls for dynamic memory allocation that detect common programming and usage errors.
Malloc(size_t size) is a system library call that (hopefully) returns a pointer to a block of memory of at least the requested size. This memory comes from a main memory resource managed by the operating system. The free(void *) function informs the operating system that you are done with a given block of dynamically-allocated memory, and that it can reclaim it for other uses.
You will use a large array to simulate main memory (static char myblock[4096]). Your malloc() function will return pointers to this large array and your free() function will let your code know that a previously-allocated region can be reclaimed and used for other purposes. Programmers can easily make some very debilitating errors when using dynamic memory. Your versions of malloc() and free() will detect these errors and will react nicely by not allowing a user to do Bad Things. Your malloc() function should use a “first free” algorithm to select blocks of memory to allocate.

How to run tests:
1. compile the memgrind.c, mymalloc.c, and mymalloc.h by type 'make' into the terminal command. The Makefile we created would take care compiling the three files.
2. After compiling, type './memgrind' into the terminal command and the code would run.
3. The code would print out which test it is running, is it mallocing or freeing (for this assignment we replace the two with: mymalloc or myfree), and the changes in memory for each of the mallocing and freeing. At the end it would show the mean time it takes for each workload in test A, B, C, D, E.

Mentionables & How it was designed:
Our code would print out the changes in memory (memory space of 4096), report any error that may appear in mallocing and freeing by identifying which test and line the error appeared. For test A and B, since we are only implementing 1 byte (A: mymalloc and myfree right after, B: mymalloc 120 times and myfree 120 times one by one) we did not have to use an array to record the memory used.

C was still similar to A and B, we are only implementing 1 bytes each time for 240 times, but this time whether we are implementing malloc or free is completely randomized. We used rand()%2 to randomly switch between mymalloc(when it is 0) and myfree(when it is 1). Test C unlike A and B, we had to use if...else to make sure there are still space (or mymalloc has yet exceeded 120 times according to instructions) to

perform mymallocing or there are pointers for myfree to free.

D was an improved version of C. Similar to C we randomly implement to mymalloc and myfree (except we are doing it 120 times), but unlike C, D mymallocs random numbers between 1 to 100. Thus we cannot set a if statement that mymallocing cannot exceed 120 times, since mymallocing would not be implementing just 1 byte. So to successfully remove the correct memory when we are freeing points, we created a memory byte array to record the memory we had used. When myfree is implemented it would free the oldest pointer thus the oldest memory from the array. Of course, if...else statements are used to make sure there is still memory space for mymalloc and if there are pointers for myfree to free.

E was different from testA,B,C,D. For this test, we did not randomly implement mymalloc or myfree (like C and D), nor did we just mymalloc and myfree 1 byte in each loops (like A,B,C).For E, we started with mymalloc and implement 1 byte in the beginning. E will continue mymallocing and add 23 bytes to the previous byte, until the memory limit (which is 4096) is reached or would be exceeded next loop. When this happens, E would switch to myfree. Myfree will continue to free the oldest pointest in each loop, until no more pointers can be freed. Then E would switch back to mymalloc and repeat the previously described process. This will continue until we have mymalloced and myfreed a total of 120 times.