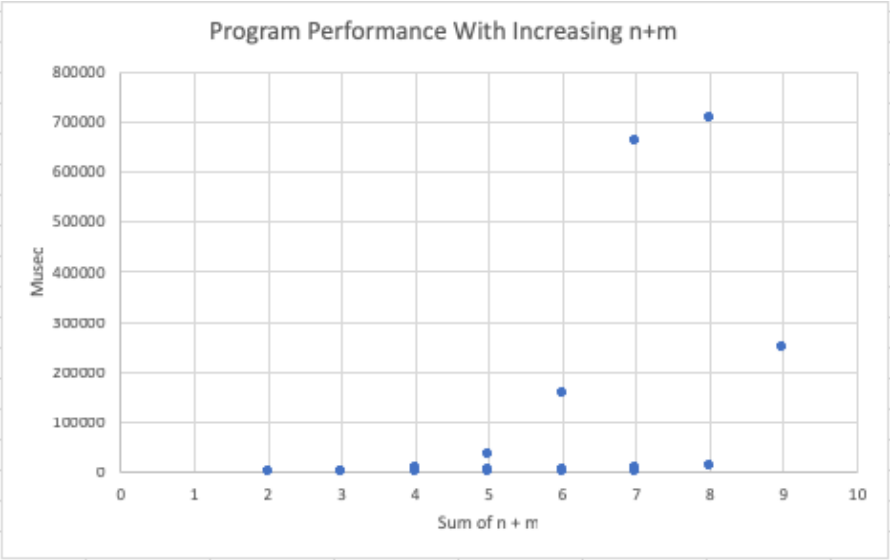
Cody Williams
CSCE 313-506

PA1 Report

The goal of programming assignment one was to become familiar with the implementation and functionality of dynamic memory allocation. The primary components of the dynamic memory allocator are the alloc function and the free function. These two functions manipulate a vector of linked list pointers called “free_list” which is used to keep track of which sections of memory have been allocated and which sections remain available for allocation.

After successfully implementing and testing my dynamic memory allocator, the common trend for performance of this program is an increased time of execution for an increased number of n and m values. Below are graphs showing this trend of decreasing program performance for increasing n and m values. Based on the data, the program performance “bottlenecks” at n values of 3 and m values of 3 and up.

n	m	musec
1	1	1543
1	2	458
1	3	411
1	4	740
1	5	423
1	6	415
2	1	566
2	2	3037
2	3	5836
2	4	3586
2	5	8975
2	6	10860
3	1	6304
3	2	36431
3	3	158515
3	4	659938
3	5	705973
3	6	248657



To improve the performance of this dynamic memory allocator implementation, it may be beneficial to introduce some kind of process forking when allocating memory at higher values of n and m. For example a condition can be included within main function of the program that ensures that every time an n value greater than one is requested at the execution of the program, an additional n-1 processes are created to allocate memory at an entirely different location within the address space. This could potentially improve the run time of the dynamic memory allocator because having multiple processes running instead of a single process would decrease the work load of any single process involved in the program. Additionally, the introduction of multiple process forks to the dynamic memory allocator would be an example of multiprogramming, which by nature, would improve the performance of the system.