Shaan Prabakar sp1311
Bhavya Patel brp93

The code my mymalloc.c consisted of 3 main parts in the code. The free,malloc and merge function. We initialized our memory as an array then pointed a node struct to this array so we can parse through it as a struct. Our struct consisted of two parameters. The first one is the size and the second one is if it is free. 0 means it is not free 1 means it is free. We do not need to store the next node because we could just parse through the array if we know the size of the node within the array we are trying to parse through. Using this knowledge we parsed through the array and assigned values to the nodes as well as splitting them based on size.

Our free function was simple besides the merge part. If we assign the inputted pointer we can refer to this as a node and modify it they way we want it. Once we do that we call the merge function. The merge function iterates through the memory array from the beginning till it finds adjacent free nodes and then it merges it.

The times we had for each test case were:
Time for A in microseconds: 273 microseconds
Time for B in microseconds: 10047 microseconds
Time for C in microseconds: 1112 microseconds
Time for D in microseconds: 2447 microseconds
Time for E in microseconds: 10129 microseconds
Time for F in microseconds: 25252 microseconds

A is obviously the shorted because it goes through one for loop of 150. B is longer than A because it is a less efficient way to do what A is doing. It does what A is doing in 2 loops. C is a loop of 150 again but its longer than A because it is more random so parsing through the array is longer. D is basically the same thing as C but with even more randomness causing more variability and length. E is about the same as B because the complexity and structure of the code is the same. F is longer than E because there is 150 more mallocs and frees.