```
/* CS 214: Systems Programming Spring 2018
 *  Professor Francisco
 *  Assignment 1: To Build A Better Malloc
 *  Richard Farrington netid: raf172
 *  Anirudh Tunoori netid: at813
 */
```

Mean Workload Runtimes:
A: 10 microseconds
B: 80 microseconds
C: 90 microseconds
D: ---
E: 500000 microseconds
F: 1100 microseconds

Design and Implementation:

Our memgrind.c contains our implementations of workloads A-D and two additional workloads. Our mymalloc.c contains two functions; our implementations of malloc and free. Both functions pass on the filename and line numbers whenever malloc or free are used improperly via one of the common errors. While the static char array[5000] holds the memory that is used, we used a char and a short as metadata. The char is set to be either ‘y’ or ‘n’ indicating if a particular block of memory is available (respectively) to use. While the short holds the number of allocated blocks in the area of the memory location. Each call by malloc() returns a pointer to a size/block specified. Within the workloads we used a random number generator for workloads C and D to determine whether to malloc() or free(), to measure the mean runtime of each workload, we used a timeval struct and incremented the runtime of each workload into a total for each cycle and took the mean at the end. For a magoirty of the workloads our implementations involved the use of an array of pointers.

Findings/ Observations:

We found that earlier workloads are given a slight precedence with respect to runtime. We also found that the total runtimes get progressively larger such that the mean runtimes in a 10-run cycle are less than in a 100-run cycle. We found that there are fluctuations and variations in the mean runtime that is perhaps dependent on processes being run simultaneously. Finally, we found similarities in the runtimes for some of the workloads even though they accomplish different tasks (B and C in particular).