Malloc

Executive Summary:

The program designed is my own implementation of malloc in C. The program uses first fit, next fit, best fit, and worst fit algorithms to test correct allocation. This specific version of malloc that is implemented keeps a count of statistics including: mallocs, frees, reuses, grows, splits, coalesces, blocks, total size requested, and max heap. This version of malloc also implements calloc and realloc. The eight tests provided for the assignment were used during debug of this malloc and 4 additional tests were created to further extend the testing this version of malloc. The statistics gathered for this assignment were collected from the four tests created.

Algorithms implemented:

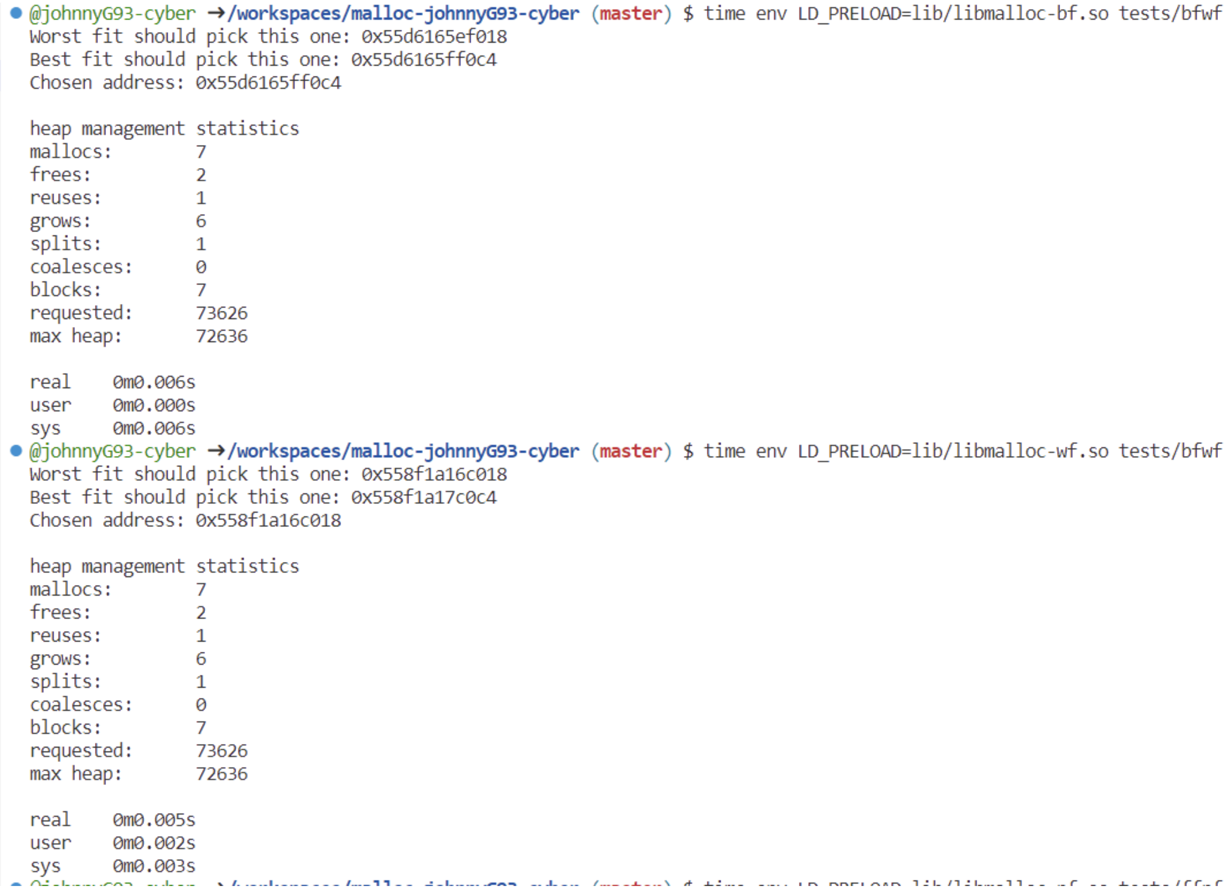
The algorithms implemented were first fit, next fit, best fit, and worst fit. The first fit algorithm is the allocation of memory in the first block that is found that is big enough to hold it and is free. The next fit algorithm is essentially the same as the first fit, however when the user asks for more memory, next fit starts at the last allocation. The best fit algorithm finds the block with the size closest to the requested size. This method is used to waste the least amount of memory. The worst fit algorithm does the opposite of best fit and finds the block that wastes the most amount of memory.

Test Implementation: The four tests created were an extension of the debug tests that were provided. In the tests I created I simply made more mallocs and frees attempting to further stress my malloc. The tests will also demonstrate the techniques used to create the malloc such as splitting and reusing blocks, and coalescing. I also tested the performance for my malloc and the system malloc for all the tests I created.

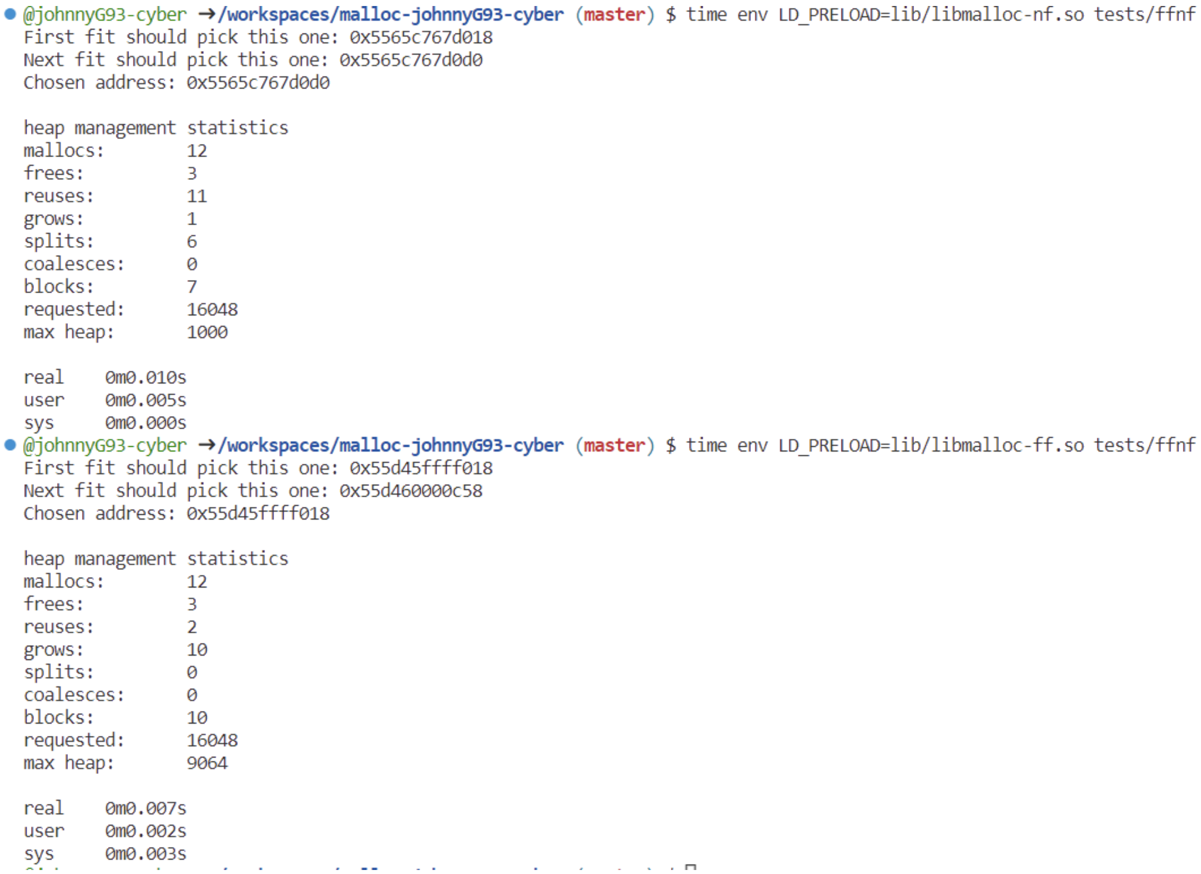
Interpretation of results:

The first fit algorithm was faster than next fit even though theoretically I believed next fit to be faster. I believe this is due to the looping one must implement to find the last allocated value. Worst fit was also faster than best fit but they were pretty close and I think this has more to do with the test cases than the actual algorithms. I also tested the times for system malloc for best fit and next fit and compared the times with my version of malloc.

Test Results for my malloc:

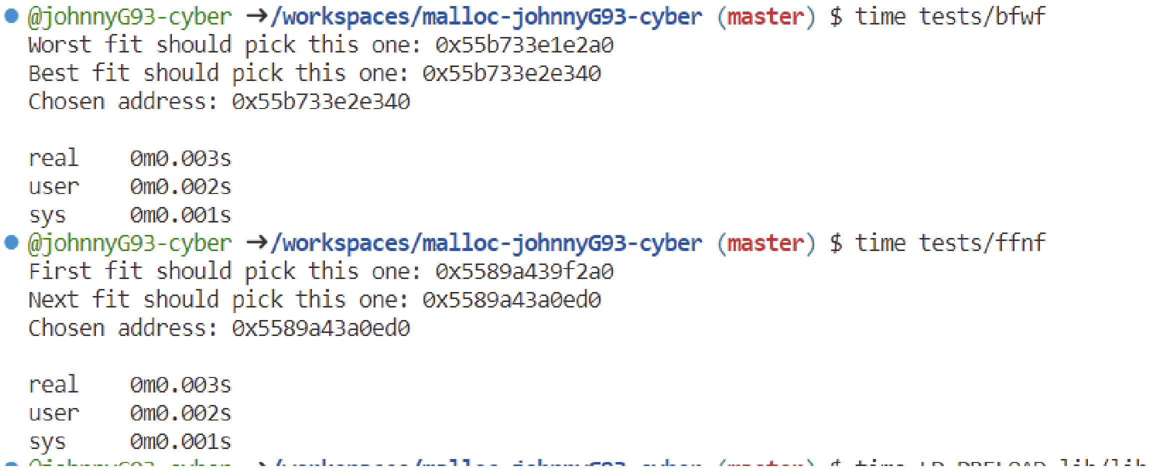


```
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-bf.so tests/bfwf
  Worst fit should pick this one: 0x55d6165ef018
  Best fit should pick this one: 0x55d6165ff0c4
  Chosen address: 0x55d6165ff0c4

  heap management statistics
  mallocs:        7
  frees:          2
  reuses:         1
  grows:          6
  splits:         1
  coalesces:      0
  blocks:         7
  requested:      73626
  max heap:       72636

  real    0m0.006s
  user    0m0.000s
  sys     0m0.006s
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-wf.so tests/bfwf
  Worst fit should pick this one: 0x558f1a16c018
  Best fit should pick this one: 0x558f1a17c0c4
  Chosen address: 0x558f1a16c018

  heap management statistics
  mallocs:        7
  frees:          2
  reuses:         1
  grows:          6
  splits:         1
  coalesces:      0
  blocks:         7
  requested:      73626
  max heap:       72636

  real    0m0.005s
  user    0m0.002s
  sys     0m0.003s
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-nf.so tests/ffnf
```



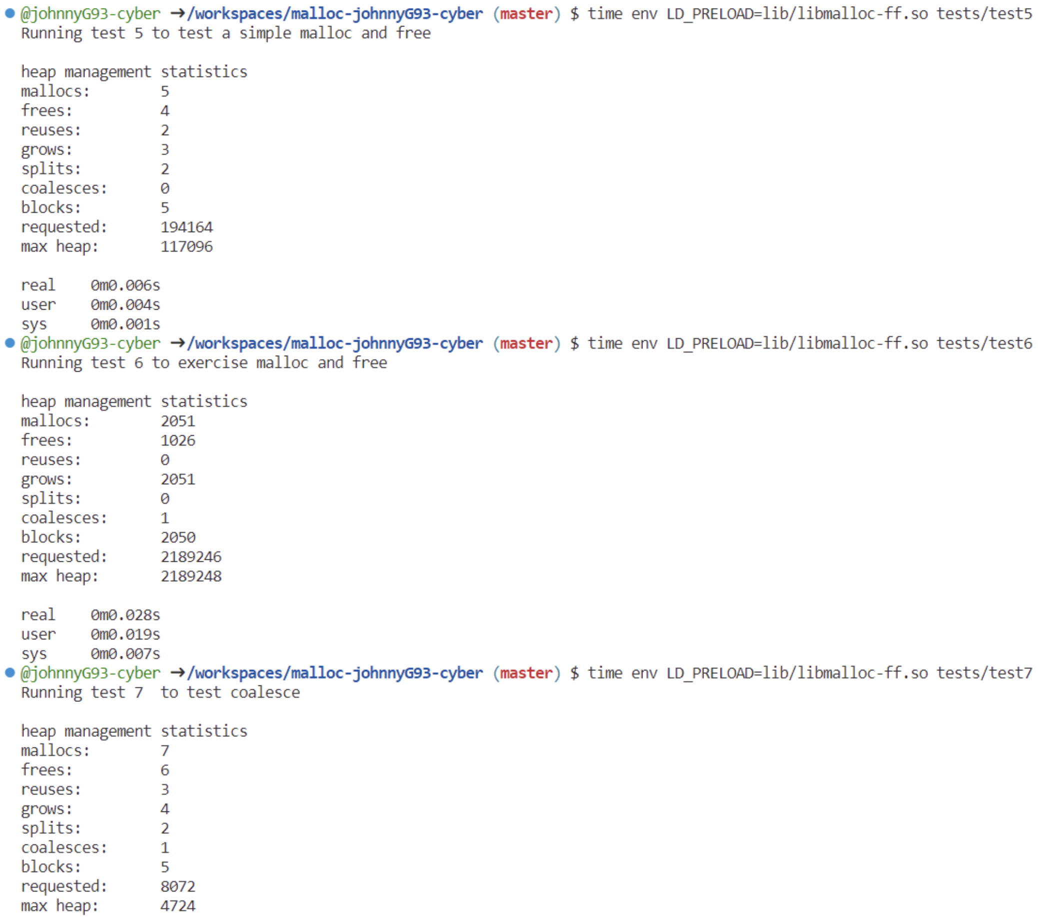
```
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-nf.so tests/ffnf
  First fit should pick this one: 0x5565c767d018
  Next fit should pick this one: 0x5565c767d0d0
  Chosen address: 0x5565c767d0d0

  heap management statistics
  mallocs:        12
  frees:          3
  reuses:         11
  grows:          1
  splits:         6
  coalesces:      0
  blocks:         7
  requested:      16048
  max heap:       1000

  real    0m0.010s
  user    0m0.005s
  sys     0m0.000s
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-ff.so tests/ffnf
  First fit should pick this one: 0x55d45ffff018
  Next fit should pick this one: 0x55d460000c58
  Chosen address: 0x55d45ffff018

  heap management statistics
  mallocs:        12
  frees:          3
  reuses:         2
  grows:          10
  splits:         0
  coalesces:      0
  blocks:         10
  requested:      16048
  max heap:       9064

  real    0m0.007s
  user    0m0.002s
  sys     0m0.003s
```

Test Results for system malloc:

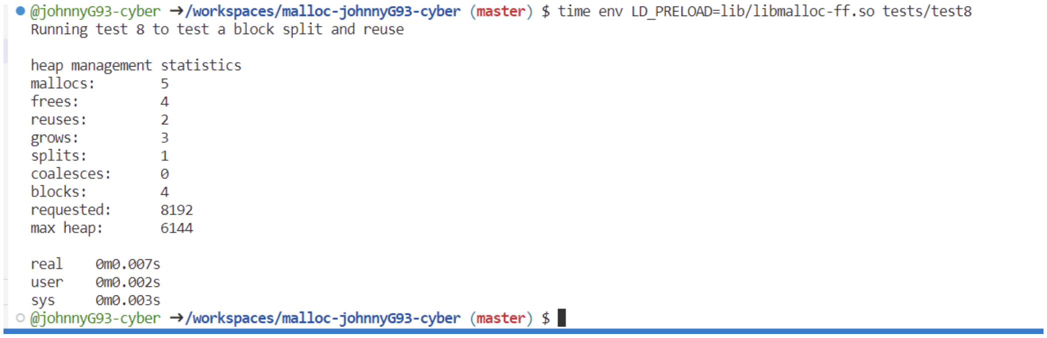


```
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time tests/bfwf
  Worst fit should pick this one: 0x55b733e1e2a0
  Best fit should pick this one: 0x55b733e2e340
  Chosen address: 0x55b733e2e340

  real    0m0.003s
  user    0m0.002s
  sys     0m0.001s
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time tests/ffnf
  First fit should pick this one: 0x5589a439f2a0
  Next fit should pick this one: 0x5589a43a0ed0
  Chosen address: 0x5589a43a0ed0

  real    0m0.003s
  user    0m0.002s
  sys     0m0.001s
```

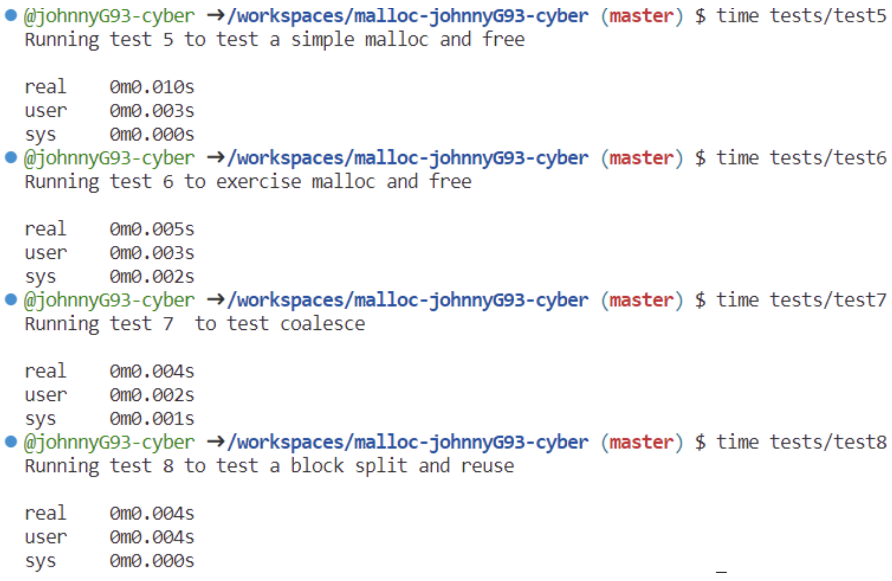
Test Results for Test 5-8:



```
● @johnnyG93-cyber ➜ /workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-ff.so tests/test5
  Running test 5 to test a simple malloc and free

  heap management statistics
  mallocs:        5
  frees:          4
  reuses:         2
  grows:          3
  splits:         2
  coalesces:      0
  blocks:         5
  requested:      194164
  max heap:       117096

  real    0m0.006s
  user    0m0.004s
  sys     0m0.001s
● @johnnyG93-cyber ➜ /workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-ff.so tests/test6
  Running test 6 to exercise malloc and free

  heap management statistics
  mallocs:        2051
  frees:          1026
  reuses:         0
  grows:          2051
  splits:         0
  coalesces:      1
  blocks:         2050
  requested:      2189246
  max heap:       2189248

  real    0m0.028s
  user    0m0.019s
  sys     0m0.007s
● @johnnyG93-cyber ➜ /workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-ff.so tests/test7
  Running test 7  to test coalesce

  heap management statistics
  mallocs:        7
  frees:          6
  reuses:         3
  grows:          4
  splits:         2
  coalesces:      1
  blocks:         5
  requested:      8072
  max heap:       4724
```



```
● @johnnyG93-cyber ➜ /workspaces/malloc-johnnyG93-cyber (master) $ time env LD_PRELOAD=lib/libmalloc-ff.so tests/test8
  Running test 8 to test a block split and reuse

  heap management statistics
  mallocs:        5
  frees:          4
  reuses:         2
  grows:          3
  splits:         1
  coalesces:      0
  blocks:         4
  requested:      8192
  max heap:       6144

  real    0m0.007s
  user    0m0.002s
  sys     0m0.003s
○ @johnnyG93-cyber ➜ /workspaces/malloc-johnnyG93-cyber (master) $ ▌
```



```
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time tests/test5
  Running test 5 to test a simple malloc and free

  real    0m0.010s
  user    0m0.003s
  sys     0m0.000s
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time tests/test6
  Running test 6 to exercise malloc and free

  real    0m0.005s
  user    0m0.003s
  sys     0m0.002s
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time tests/test7
  Running test 7  to test coalesce

  real    0m0.004s
  user    0m0.002s
  sys     0m0.001s
● @johnnyG93-cyber →/workspaces/malloc-johnnyG93-cyber (master) $ time tests/test8
  Running test 8 to test a block split and reuse

  real    0m0.004s
  user    0m0.004s
  sys     0m0.000s
```

Conclusion:

It is possible that best fit is faster than worst fit and next fit is faster than first fit since the system malloc chose those algorithms to malloc. For my malloc, this was not the case probably due to the implementation of the algorithms. I also found that the system malloc is significantly faster than the malloc I implemented.