Executive Summary

For this coding assignment malloc and free functions are implemented, apart from using the system’s functions. Moreover, allocation of the memory block is needed to do with four different types of algorithms. The algorithms are first fit, next fit, best fit, and worst fit. These four ways set new memory in different forms which should affect code utilization depending on the input. A heap is created to keep track of allocated memory and it also interacts with libraries for processing malloc and free. The free function has the job combining block with checking requirements. Firstly, the free function checks if the block following a free block is free. If it is, the function combines them, making them one. The malloc function also holds a memory block splitting code which separates free space from an occupied memory block. With the malloc, calloc and realloc is being simply implied.

Description of the Algorithms

First Fit – First fit starts at the very beginning of the heap and keeps rotating through the allocated memory in search of required space. In this search, the algorithm does not care how much additional space is left after aligning. After getting to fulfill required memory space, the algorithm exits and returns the address where it needs to be used.

Next Fit – Next fit starts where the last allocation occurred which can in middle or very end. The algorithm moves to the next memory block in heap for looking required room of its requirement. If the algorithm reaches end and does not find a memory block and faces NULL, algorithm will continue from the beginning of the heap looking for room. If room found then return with that address, else look till last allocation address and return NULL. Returning NULL signifies the need of growing heap for the allocation since no block memory meets the storage requirement.

Best Fit – Best fit starts from the top of heap and goes thoroughly till it reaches end and faces NULL. While it irritates, the algorithm checks with block memory if it meets the requirement. Even after finding a block, it keeps going with the goal of leaving the least free space. For example, after algorithm have irritated thoroughly, it gets memory of 100, 200, and 300 and the requirement is only of 50. The algorithm will pick the address of 100 memory block because it has the least remaining memory left, which will be 50. If went with 200 or 300, left memory would be 150 or 250, respectively, which will be more than 50.

Worst Fit – Worst is just the opposite of Best Fit. The worst fit starts from the top of heap and goes thoroughly till it reaches end and faces NULL. While it irritates, the algorithm checks with block memory if it meets the requirement. Even after finding a block, it keeps going with the goal of leaving the most free space. For example, after algorithm have irritated thoroughly, it gets memory of 100, 200, and 300 and the requirement is only of 50. The algorithm will pick the address of 300 memory block because it has the most remaining memory left, which will be 250. If went with 100 or 200, the left memory would be 50 or 150, respectively, which will be less than 250.

Test Implementation

Firstly, to be able to make implied malloc to execute "env LD_PRELOAD=lib/libmalloc-ff.so tests/ffnf" is run. This stops the system malloc to executes. As there are eight tests for checking the code, each of them implies to different parts. Test 1 executes for checking a simple memory allocation is done with freeing it immediately. This does not cover splitting or much, just checking for any error in the functions. Test 2, check through by creating a two-dimensional array of char and assigning memory while freeing in between. Test 3 covers coalesce with freeing strings one after another. Test 4 goes over splitting and reusing memory by assigning and freeing right after, then doing same with new variable. For basic test for calloc and realloc, the tests realloc and calloc check it by performing simple code. Test 5, 6, 7, and 8 are personally implied. All deals with random allocation between code and freeing, even dealing with calloc and malloc. For example, allocation only some from an char array and only freeing random one by for loop. This makes code verification better.

Test Results

	First Fit	Next Fit
First fit should pick this one:	0x56222ea3b018	0x5605e871c018
Next fit should pick this one:	0x56222ea3cc58	0x5605e871dc58
Chosen address:	0x56222ea3b018	0x5605e871dc58
heap management statistics		
mallocs	12	12
frees	3	3
reuses	12	12
grows	10	10
splits	0	0
coalesces	0	0
blocks	9	9
requested	16048	16048
max heap	9064	9064
Local malloc time	0.01	0.004

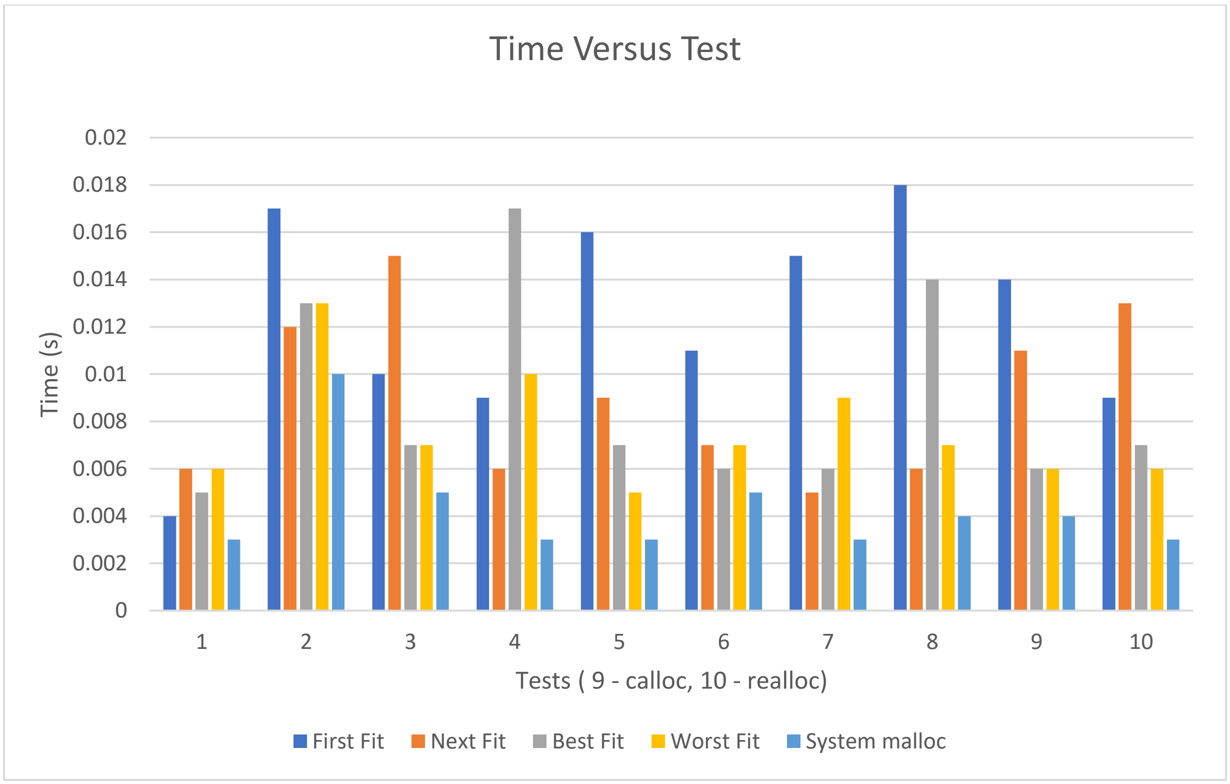
First Fit and Next Fit Allocation with Statistics

	Best Fit	Worst Fit
Worst fit should pick this one:	0x55c6923ee018	0x55c6923ee018
Best fit should pick this one:	0x55c6923fe0c4	0x55c6923fe0c4
Chosen address:	0x55c6923fe0c4	0x55c6923ee018
heap management statistics		
mallocs	7	7
frees	2	2
reuses	7	7
grows	6	6
splits	1	1
coalesces	0	0
blocks	6	6
requested	73626	73626
max heap	72636	72636
local malloc time	0.006	0.004

Best Fit and Worst Fit Allocation with Statistics

	test1	test2	test3	test4	test5	test6	test7	test8	calloc	realloc
mallocs	2	1027	4	3	43	4	42	82	2	3
frees	1	514	3	2	22	3	9	65	1	1
reuses	2	1027	4	3	43	4	42	82	2	3
grows	2	1026	3	2	43	3	42	81	2	3
splits	0	1	1	1	0	1	0	1	0	0
coalesces	0	2	2	1	1	2	1	2	0	0
blocks	1	1024	1	1	41	1	40	79	1	2
requested	66559	1180670	5472	4096	6064	3554	3648	13248	1048	1056
max heap	66560	1115136	3424	3072	6064	3232	3648	8328	1044	1044

Statistics of all Tests



Time comparison between all local malloc algorithms and system malloc with respect to all tests.

First Fit Time (s)	0.004	0.017	0.01	0.009	0.016	0.011	0.015	0.018	0.014	0.009
Next Fit Time (s)	0.006	0.012	0.015	0.006	0.009	0.007	0.005	0.006	0.011	0.013
Best Fit Time (s)	0.005	0.013	0.007	0.017	0.007	0.006	0.006	0.014	0.006	0.007
Worst Fit Time (s)	0.006	0.013	0.007	0.01	0.005	0.007	0.009	0.007	0.006	0.006
System malloc Time (s)	0.003	0.01	0.005	0.003	0.003	0.005	0.003	0.004	0.004	0.003

Actual value of time testing between all local malloc algorithms and system malloc with respect to all tests.

Result Description

As seen from the results, system malloc proves to be faster in every one of the test cases compared with any of the algorithms. Looking at algorithms, all seem to address at correct values. However there seem to be problems with statistics as it does not change per the algorithm change. There seems to be a need for depth increment for it. Talking about test cases, most of them seem to be outputting correct requested and max heap. However, additional malloc comes in that is due to system trying to call to system malloc which also leads to extra allocations for some test cases.

Conclusion

In this programming task, we created our own versions of malloc and free functions, in addition to using the built-in ones provided by the system. Four different allocation algorithms, which are first fit, next fit, best fit, and worst fit, need to be implemented, each of which allocates memory blocks in a different way, potentially affecting the efficiency of the code depending on the input. To keep track of allocated memory, we will create a heap that interacts with the libraries for processing malloc and free. The free function will check if the block following a freed block is also free, and if so, combine them into a single block. The malloc function will also have code for splitting memory blocks into free space and occupied memory. The functions calloc and realloc will also be implemented.