

Section 1

Executive Summary

This report have details for the implementation and evaluation of a custom memory management library designed to mimic standard malloc, free, calloc, and realloc functionalities. The project will compare the performance of four memory allocation strategies: First Fit, Best Fit, Worst Fit, and Next Fit, alongside the default system malloc. The analysis focuses on allocation speed, memory fragmentation, and resource utilization. Key findings indicate trade-offs between efficiency and fragmentation across different allocation strategies, with detailed results presented for various test cases.

Section 2

Description of the algorithms implemented

First Fit: Iterates from the start of the heap to locate the first free block that meets the requested size.

Best Fit: Searches for the smallest free block that accommodates the request.

Worst Fit: Allocates the largest free block available, leaving larger remainders for future allocations.

Next Fit: Continues the search for a free block from the last allocated block, wrapping around to the beginning if necessary.

System malloc: It is the standard library implementation, used as a baseline for comparison.

Section 3

Test Implementation/ results

Next Fit

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-  
nf.so tests/ffnf
```

First fit should pick this one: 0x61aa0ed7c020

Next fit should pick this one: 0x61aa0ed7dca0

Chosen address: 0x61aa0ed7dca0

heap management statistics

mallocs: 12

frees: 3

reuses: 2

grows: 10

splits: 0

coalesces: 0

blocks: 10

requested: 16064

max heap: 9384

total malloc time (seconds): 0.000029

////////////////////////////////////

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-  
nf.so tests/calloc
```

calloc test PASSED

heap management statistics

mallocs: 2

frees: 1

reuses: 0

grows: 2

splits: 0

coalesces: 0

blocks: 2

requested: 1044

max heap: 1108

total malloc time (seconds): 0.000010

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-nf.so tests/realloc

realloc test PASSED

heap management statistics

mallocs: 3

frees: 1

reuses: 0

grows: 3

splits: 0

coalesces: 0

blocks: 3

requested: 1044

max heap: 1140

total malloc time (seconds): 0.000012

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-nf.so tests/test1

Running test 1 to test a simple malloc and free

heap management statistics

mallocs: 2

frees: 1

reuses: 0
grows: 2
splits: 0
coalesces: 0
blocks: 2
requested: 66560
max heap: 66624
total malloc time (seconds): 0.000017

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-nf.so tests/test2

Running test 2 to exercise malloc and free

heap management statistics

mallocs: 1027
frees: 514
reuses: 1
grows: 1026
splits: 0
coalesces: 1
blocks: 1025
requested: 1180672
max heap: 1147968
total malloc time (seconds): 0.003391

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-nf.so tests/test3

Running test 3 to test coalesce

heap management statistics

mallocs: 4

frees: 3

reuses: 0

grows: 4

splits: 0

coalesces: 0

blocks: 4

requested: 5472

max heap: 5600

total malloc time (seconds): 0.000016

////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-nf.so tests/test4

Running test 4 to test a block split and reuse

heap management statistics

mallocs: 3

frees: 2

reuses: 1

grows: 2

splits: 1

coalesces: 1

blocks: 2

requested: 4096

max heap: 3136

total malloc time (seconds): 0.000011

////////////////////////////////

First Fit

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-ff.so tests/ffnf
```

First fit should pick this one: 0x5a459118c020

Next fit should pick this one: 0x5a459118dca0

Chosen address: 0x5a459118c020

heap management statistics

mallocs: 12

frees: 3

reuses: 2

grows: 10

splits: 0

coalesces: 0

blocks: 10

requested: 16064

max heap: 9384

total malloc time (seconds): 0.000028

////////////////////////////////////

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-ff.so tests/calloc
```

calloc test PASSED

heap management statistics

mallocs: 2

frees: 1

reuses: 0

grows: 2

splits: 0
coalesces: 0
blocks: 2
requested: 1044
max heap: 1108
total malloc time (seconds): 0.000010

////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-ff.so tests/realloc

realloc test PASSED

heap management statistics

mallocs: 3
frees: 1
reuses: 0
grows: 3
splits: 0
coalesces: 0
blocks: 3
requested: 1044
max heap: 1140
total malloc time (seconds): 0.000012

////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-ff.so tests/test1

Running test 1 to test a simple malloc and free

heap management statistics

mallocs: 2

```
frees:    1
reuses:   0
grows:    2
splits:   0
coalesces: 0
blocks:   2
requested: 66560
max heap:  66624
total malloc time (seconds): 0.000013
```

```
//////////
```

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-ff.so tests/test2
```

Running test 2 to exercise malloc and free

heap management statistics

```
mallocs:   1027
frees:     514
reuses:    1
grows:    1026
splits:    0
coalesces: 1
blocks:   1025
requested: 1180672
max heap:  1147968
total malloc time (seconds): 0.003154
```

```
//////////
```

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-ff.so tests/test3
```

Running test 3 to test coalesce

heap management statistics

mallocs: 4

frees: 3

reuses: 0

grows: 4

splits: 0

coalesces: 0

blocks: 4

requested: 5472

max heap: 5600

total malloc time (seconds): 0.000017

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-ff.so tests/test4

Running test 4 to test a block split and reuse

heap management statistics

mallocs: 3

frees: 2

reuses: 1

grows: 2

splits: 1

coalesces: 1

blocks: 2

requested: 4096

max heap: 3136

total malloc time (seconds): 0.000012

////////////////////////////////////

Best Fit

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-bf.so tests/bfwf
```

Worst fit should pick this one: 0x5e797210e020

Best fit should pick this one: 0x5e797211e0ec

Chosen address: 0x5e797211e0ec

heap management statistics

mallocs: 7

frees: 2

reuses: 1

grows: 6

splits: 1

coalesces: 0

blocks: 7

requested: 73636

max heap: 72828

total malloc time (seconds): 0.000021

////////////////////////////////////

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-bf.so tests/calloc
```

calloc test PASSED

heap management statistics

mallocs: 2

frees: 1

reuses: 0

grows: 2

splits: 0
coalesces: 0
blocks: 2
requested: 1044
max heap: 1108
total malloc time (seconds): 0.000010

////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-bf.so tests/realloc

realloc test PASSED

heap management statistics

mallocs: 3
frees: 1
reuses: 0
grows: 3
splits: 0
coalesces: 0
blocks: 3
requested: 1044
max heap: 1140
total malloc time (seconds): 0.000015

////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-bf.so tests/test1

Running test 1 to test a simple malloc and free

heap management statistics

mallocs: 2

```
frees:    1
reuses:   0
grows:    2
splits:   0
coalesces: 0
blocks:   2
requested: 66560
max heap:  66624
total malloc time (seconds): 0.000013
```

```
////////////////////////////////
```

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-
bf.so tests/test2
```

Running test 2 to exercise malloc and free

heap management statistics

```
mallocs:   1027
frees:     514
reuses:    1
grows:     1026
splits:    0
coalesces: 1
blocks:    1025
requested: 1180672
max heap:  1147968
total malloc time (seconds): 0.003280
```

```
////////////////////////////////
```

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-
bf.so tests/test3
```

Running test 3 to test coalesce

heap management statistics

mallocs: 4

frees: 3

reuses: 0

grows: 4

splits: 0

coalesces: 0

blocks: 4

requested: 5472

max heap: 5600

total malloc time (seconds): 0.000015

////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-bf.so tests/test4

Running test 4 to test a block split and reuse

heap management statistics

mallocs: 3

frees: 2

reuses: 1

grows: 2

splits: 1

coalesces: 1

blocks: 2

requested: 4096

max heap: 3136

total malloc time (seconds): 0.000011

////////////////////////////////

Worst Fit

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-wf.so tests/bfwf
```

Worst fit should pick this one: 0x5cd8d196f020

Best fit should pick this one: 0x5cd8d197f0ec

Chosen address: 0x5cd8d196f020

heap management statistics

mallocs: 7

frees: 2

reuses: 1

grows: 6

splits: 1

coalesces: 0

blocks: 7

requested: 73636

max heap: 72828

total malloc time (seconds): 0.000023

////////////////////////////////

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-wf.so tests/calloc
```

calloc test PASSED

heap management statistics

mallocs: 2

frees: 1

reuses: 0

grows: 2

splits: 0
coalesces: 0
blocks: 2
requested: 1044
max heap: 1108
total malloc time (seconds): 0.000013

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-wf.so tests/realloc

realloc test PASSED

heap management statistics

mallocs: 3
frees: 1
reuses: 0
grows: 3
splits: 0
coalesces: 0
blocks: 3
requested: 1044
max heap: 1140
total malloc time (seconds): 0.000011

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-wf.so tests/test1

Running test 1 to test a simple malloc and free

heap management statistics

mallocs: 2

```
frees:    1
reuses:   0
grows:    2
splits:   0
coalesces: 0
blocks:   2
requested: 66560
max heap:  66624
total malloc time (seconds): 0.000015
```

```
//////////
```

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-
wf.so tests/test2
```

Running test 2 to exercise malloc and free

heap management statistics

```
mallocs:   1027
frees:     514
reuses:    1
grows:     1026
splits:    0
coalesces: 1
blocks:    1025
requested: 1180672
max heap:  1147968
total malloc time (seconds): 0.003176
```

```
//////////
```

```
@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) $ env LD_PRELOAD=lib/libmalloc-
wf.so tests/test3
```

Running test 3 to test coalesce

heap management statistics

mallocs: 4

frees: 3

reuses: 0

grows: 4

splits: 0

coalesces: 0

blocks: 4

requested: 5472

max heap: 5600

total malloc time (seconds): 0.000018

////////////////////////////////////

@nooraldeen00 → /workspaces/malloc-nooraldeen00 (master) \$ env LD_PRELOAD=lib/libmalloc-wf.so tests/test4

Running test 4 to test a block split and reuse

heap management statistics

mallocs: 3

frees: 2

reuses: 1

grows: 2

splits: 1

coalesces: 1

blocks: 2

requested: 4096

max heap: 3136

total malloc time (seconds): 0.000010

////////////////////////////////////

Malloc

@nooraldeen00 → /workspaces/malloc-nooraldeen00/tests (master) \$ gcc malloc_test.c

@nooraldeen00 → /workspaces/malloc-nooraldeen00/tests (master) \$./a.out

Starting malloc test...

Allocating 1000 blocks of size 128 bytes...

Memory allocation successful.

Reallocating every 10th block to 256 bytes...

Memory reallocation successful.

Freeing all allocated memory...

Memory free successful.

Allocating 1000 blocks of size 128 bytes again...

Second memory allocation successful.

Test completed.

Performance

The standard malloc() demonstrates the best performance with the lowest allocation time (0.000020 seconds) due to its optimized design for fast lookups and minimal overhead. Best Fit (0.000021 seconds) performed the best, balancing search overhead and allocation efficiency. Next Fit (0.000029 seconds) and First Fit (0.000028 seconds) showed comparable results, with slightly slower times due to their linear search approaches. Worst Fit (0.000023 seconds) was slightly less efficient than Best Fit but performed better than Next Fit and First Fit, trading speed for reduced fragmentation by selecting larger blocks. Overall, malloc() remains the fastest, while Best Fit emerged as the most efficient among the custom implementations.

Explanation and Interpretation of Results

Malloc() consistently outperforms custom implementations due to its optimized design. So, Best Fit demonstrated the best balance between speed and fragmentation by minimizing splits, while Worst Fit showed moderate performance, prioritizing reduced fragmentation. Next Fit and First Fit performed similarly, with slower allocation times due to their sequential search methods. An anomaly observed was that Worst Fit occasionally led to larger max heap sizes despite fewer splits.

Conclusion on AI performance.

The AI assistant was helpful in implementing the code structure and providing guidance on the allocator logic, making the process faster and more organized. However, it did not produce accurate results for some statistics and encountered issues such as segmentation faults during execution. While the AI excelled in explaining concepts and creating a framework, it failed in ensuring complete functionality and error-free implementation. Despite these challenges, the process provided valuable learning, though it helps a lot. I learned more By encountering issues like inaccurate statistics and segmentation faults, I learned more about debugging, memory management intricacies, and the importance of handling edge cases in code.