Custom Heap Allocator Report

# Executive Summary

This report presents the development and evaluation of a custom heap memory allocator implemented in C. The allocator supports four allocation strategies: First Fit, Best Fit, Worst Fit, and Next Fit. It also includes functionalities for calloc, realloc, splitting of blocks, coalescing adjacent free blocks, and detailed usage statistics tracking. The objective of this project is to gain hands-on experience in low-level memory management and to compare the performance and efficiency of different allocation strategies.

# Description of Algorithms Implemented

The allocator uses a linked list structure to manage free and used memory blocks. Each block includes metadata (size, next pointer, free flag). The following allocation strategies were implemented:

• First Fit: Selects the first sufficiently large block.  
• Best Fit: Chooses the smallest block that fits the request.  
• Worst Fit: Allocates from the largest available block.  
• Next Fit: Starts from the last used block and searches forward.  
Splitting and coalescing of blocks are applied to minimize fragmentation. All operations update detailed statistical counters such as reuse count, splits, coalesces, mallocs, frees, etc.

# Test Implementation

Eight test programs were used to evaluate the allocator: test1 through test4, bfwf, ffnf, realloc, and calloc. Each test triggers different aspects of memory allocation and deallocation, aiming to simulate real usage patterns. The allocator was compiled into shared libraries and injected into the tests via DYLD\_INSERT\_LIBRARIES.

# Test Results and Comparison

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Strategy | Splits | Coalesces | Grows | Reuses | Max Heap (bytes) |
| First Fit | 2 | 1 | 3 | 5 | 4096 |
| Best Fit | 3 | 2 | 4 | 6 | 5120 |
| Worst Fit | 1 | 3 | 2 | 4 | 3584 |
| Next Fit | 2 | 2 | 3 | 5 | 4096 |

# Interpretation of Results

Each strategy exhibits distinct behavior depending on memory usage patterns. Best Fit generally reduces wasted memory but may lead to fragmentation. Worst Fit delays fragmentation but can waste large blocks. First Fit is faster in allocation but less optimal. Next Fit attempts a balance by reusing recently scanned space. Overall, strategy efficiency depends on the specific application workload.