Test Plan for MP4

Name:

Aarav Rekhi

Objective:

The objective of this test plan is to verify the functionality and performance of MP4. The tests aim to cover a wide range of scenarios, such as special cases, boundary conditions, memory allocation patterns, behavior of the roving pointer, and detection of memory leaks. Detailed descriptions of each test case, expected outcomes, and justifications for success are provided to ensure thorough testing.

Test Cases:

1. **Boundary Conditions for Memory Block Sizes:**
    a. **Description**: Evaluate the memory allocator's behavior when handling memory blocks of various sizes, including edge cases and boundary conditions.
    b. **Steps**:
        i. Allocate memory blocks of sizes 1, 2, and 3 bytes.
        ii. Print the free list to inspect the memory allocation.
        iii. Free memory blocks 1 and 3.
        iv. Print the free list and verify the presence of a hole between blocks 1 and 3.
        v. Repeat steps 1-4 with different allocation patterns and sizes, including maximum and minimum allowed sizes.
        vi. Allocate memory to fill up a whole page (or almost a whole page, considering the header space).

vii. Free all memory from the free list and verify that the list is empty.

c. **Expected Output**: Detailed free list output showing the allocation and deallocation of memory blocks. Verification of holes between freed blocks. Empty free list after freeing all memory.

d. **Justification**: Ensures that the memory allocator correctly manages memory blocks of various sizes and handles boundary conditions without memory leaks or corruption.

2. Roving Pointer Behavior:

a. **Description**: Assess the behavior of the roving pointer, which determines the starting point for memory allocation within the free list.

b. **Steps**:

i. Allocate memory multiple times using different search policies (first-fit, best-fit, worst-fit).

ii. Call Mem_stats at the end of each allocation and verify the memory statistics.

iii. Ensure that the roving pointer spreads the allocation of memory blocks throughout the free list.

c. **Expected Output**: Spread out allocation of memory blocks throughout the free list. Consistent and accurate memory statistics reported by Mem_stats.

d. **Justification**: Verifies that the roving pointer efficiently distributes memory allocation, preventing fragmentation and improving memory utilization.

3. Detection of Memory Leaks:

a. **Description**: Check the memory allocator's ability to detect and prevent memory leaks.

b. **Steps**:

i. Allocate and free memory blocks in various patterns and sizes.
ii. Call Mem_stats after each allocation and deallocation.
iii. Verify that the total memory stored in the free list matches the number of pages requested times the page size.
iv. Verify that the message "all memory is in the heap -- no leaks are possible" is printed when there are no memory leaks detected.

c. **Expected Output**: Proper memory statistics reported by Mem_stats. Absence of memory leak message when no leaks occur.
d. **Justification**: Ensures that the memory allocator effectively manages memory resources and prevents memory leaks, which could lead to system instability or resource exhaustion.

4. Performance Testing:

a. **Description**: Evaluate the performance of the memory allocator under varying loads and conditions.
b. **Steps**:
i. Allocate and free a large number of memory blocks with different sizes and patterns.
ii. Measure the time taken for memory allocation and deallocation operations.
iii. Monitor memory usage and fragmentation using Mem_stats during and after the test.
c. **Expected Outcome**: Reasonable performance with acceptable memory utilization and low fragmentation.
d. **Justification**: Validates that the memory allocator can handle real-world workloads efficiently without significant performance degradation or memory wastage.

Conclusion:

The test plan outlined above ensures thorough testing of MP4, covering various scenarios and ensuring correctness and efficiency. By systematically executing these tests and analyzing the outcomes, any potential issues can be identified and addressed, leading to a reliable and high-performance memory management solution.