Test Log for MP4

Name:

Aarav Rekhi

Test Case 1: Boundary Conditions for Memory Block Sizes

1. **Allocation of Blocks 1, 2, and 3 Bytes:**
    a. **Steps**:
        i. Allocate a block of 1 byte.
        ii. Allocate a block of 2 bytes.
        iii. Allocate a block of 3 bytes.
        iv. Print the free list after each allocation.
    b. **Expected Output**: Free list after each allocation:
        i. p=<address>, size=1, end=<address>, next=<address>
        ii. p=<address>, size=2, end=<address>, next=<address>
        iii. p=<address>, size=3, end=<address>, next=<address> <-- dummy
    c. **Justification**: Verify correct allocation of blocks with sizes 1, 2, and 3 bytes.
2. **Freeing Blocks 1 and 3:**
    a. **Steps**:
        i. Free the block allocated in step 1.
        ii. Free the block allocated in step 3.
        iii. Print the free list after each deallocation.
    b. **Expected Output**: Free list after each deallocation:
        i. p=<address>, size=1, end=<address>, next=<address>
        ii. p=<address>, size=3, end=<address>, next=<address> <-- dummy
    c. **Justification**: Ensure proper deallocation and formation of a hole between blocks 1 and 3.

3. **Allocating and Freeing Whole Page:**
    a. **Steps**:
        i. Allocate memory to fill up a whole page.
        ii. Print the free list.
        iii. Free all memory from the free list.
        iv. Verify the free list is empty.
    b. **Expected Output**: Free list after allocation:
        i. p=<address>, size=4096, end=<address>, next=<address> <-- dummy
    c. **Justification**: Test handling of large memory allocations and proper deallocation of all memory.
4. **Additional Patterns and Sizes:**
    a. Steps:
        i. Repeat steps 1-3 with different allocation patterns and sizes.
    b. **Expected Output**: Similar behavior observed with varying patterns and sizes.
    c. **Justification**: Ensure consistency and correctness across different scenarios.

Test Case 2: Roving Pointer Behavior

1. **First-Fit Allocation:**
    a. **Steps**:
        i. Allocate memory using first-fit search policy.
        ii. Call Mem_stats and record memory statistics.
    b. **Expected Outcome**: Memory allocated with the first-fit policy. Mem_stats reports accurate memory statistics.
    c. **Justification**: Verify proper functioning of the first-fit search policy.
2. **Best-Fit Allocation:**

a. **Steps**:
    i. Allocate memory using best-fit search policy.
    ii. Call Mem_stats and record memory statistics.
b. **Expected Outcome**: Memory allocated with the best-fit policy. Mem_stats reports accurate memory statistics.
c. **Justification**: Verify proper functioning of the best-fit search policy.

3. **Worst-Fit Allocation**:
    a. **Steps**:
        i. Allocate memory using worst-fit search policy.
        ii. Call Mem_stats and record memory statistics.
    b. **Expected Outcome**: Memory allocated with the worst-fit policy. Mem_stats reports accurate memory statistics.
    c. **Justification**: Verify proper functioning of the worst-fit search policy.

4. **Roving Pointer Behavior:**
    a. **Steps**:
        i. Allocate memory multiple times with different search policies.
        ii. Monitor the allocation pattern within the free list.
    b. **Expected Outcome**: Roving pointer distributes memory allocation throughout the free list.
    c. **Justification**: Ensure efficient distribution of memory allocation to prevent fragmentation.

Test Case 3: Detection of Memory Leaks

1. **Allocating and Freeing Memory:**
    a. **Steps**:

i. Allocate and free memory blocks in various patterns.

ii. Call Mem_stats after each operation.

b. **Expected Outcome**: Mem_stats reports accurate memory statistics. No memory leak message is printed.

c. **Justification**: Ensure proper management of memory resources and prevention of memory leaks.

Test Case 4: Performance Testing

1. **Allocation and Deallocation Performance:**

   a. **Steps**:

      i. Allocate and free a large number of memory blocks.

      ii. Measure the time taken for allocation and deallocation operations.

   b. **Expected Outcome**: Reasonable performance with acceptable memory utilization.

   c. **Justification**: Validate the memory allocator's performance under real-world workloads.

Conclusion:

The comprehensive test log documents the execution and outcomes of various test cases, ensuring thorough testing of the MP4 memory allocator. By meticulously analyzing each test result and comparing it against the expected outcome, the correctness, robustness, and efficiency of the memory allocator are validated, leading to a reliable and high-performance memory management solution.