CS 5393 C++ FOR CS: LAB 3

C++ MEMORY MANAGEMENT SYSTEM

Overview: In this lab, you will design and implement a custom memory management system that utilizes the sign post methodology to handle memory allocations efficiently. You'll create a memory pool system that can coexist with standard memory operations, track allocations, manage fragmentation, and provide visualization tools for memory analysis. The system must maintain proper memory alignment for all allocations to ensure optimal performance and compatibility.

Repo Link: https://classroom.github.com/a/d2RXxZ-Q

Due: November 25, 2024 @ 11:59pm.

CODE REQUIREMENTS (70%)

1. MEMORY ALIGNMENT AND POOL SETUP (20%)

- Implement proper memory alignment for:
  - The initial memory pool allocation
  - Individual block allocations within the pool
- Handle alignment padding efficiently:
  - Minimize wasted space from alignment requirements
  - Optimize block splitting to maintain alignment
- Verify alignment correctness:
  - Runtime alignment checks (debug builds)

2. CORE MEMORY MANAGEMENT SYSTEM (25%)

- Implement a memory pool system that pre-allocates a user defined size block of memory
- Override global `new` and `delete` operators while maintaining standard allocation capabilities
- Implement tracking of filename and line number for allocations
- Create a sign post-based memory block system with headers
- Support both single object and array allocations/deallocations

3. MEMORY BLOCK MANAGEMENT (15%)

- Implement an efficient block finding algorithm (better than O(n))
- Create a block splitting system for optimal memory usage
- Implement a defragmentation system that:
  - Merges adjacent free blocks
  - Minimizes performance impact on active allocations

    - Maintains alignment requirements during merging

4. DEBUG AND VISUALIZATION FEATURES (10%)

- Implement comprehensive memory statistics tracking:
    - Current memory usage and waste due to alignment
    - Peak memory usage
    - Fragmentation percentage
    - Number of allocations/deallocations
    - Average block size
    - Alignment overhead statistics
- Create a memory visualization system that shows:
    - Memory block layout with alignment boundaries
    - Block sizes and status
    - Allocation sources (filename/line)
    - Alignment padding visualization
- Implement configurable debug logging:
    - Memory operation tracking
    - Error detection
    - Warning systems for potential issues
    - Alignment violation reporting
    - All debug features must be compile-time configurable

TEST APPLICATION REQUIREMENTS (10%)

1. Implement a test application that demonstrates:
    - Various memory/object sizes
    - High-frequency allocations/deallocations
2. Include performance benchmarks comparing:
    - Managed vs standard allocations
    - Impact of defragmentation
    - Memory usage patterns
    - Alignment overhead impact

REPORT REQUIREMENTS (20%)

1. SYSTEM DESIGN DOCUMENTATION (6%)

- Detailed explanation of the memory management architecture
- Block management strategy and implementation details
- Defragmentation system design and operation
- Alignment strategy and implementation details
- Integration guide with code examples

2. PERFORMANCE ANALYSIS (6%)

- Benchmark results and analysis
- Comparison with standard memory allocation
- Impact of different optimization strategies
- Analysis of defragmentation performance
- Alignment overhead from alignment requirements

3. VISUALIZATION DOCUMENTATION (3%)

- Description of visualization tools and features
- Screenshots and explanation of different views
- Usage guide for debugging tools
- Example scenarios with visualization interpretation

SUBMISSION REQUIREMENTS

1. All source code must be submitted via the provided GitHub repository
2. Complete technical report in PDF format
3. Test application source code and results
4. Any additional tools or scripts used for visualization
5. You can update the compile command in the make file to ensure it will properly compile your code. You can add as many additional make commands as desired, be sure any additional make commands are documented in the report above.

If your project fails the compile action upon final push to git repo, it will receive a 0. Be sure to resolve any issues with the compile action prior to deadline.