**Task 1 Documentation Outline**

**Title**: Task 1 - Implementing and Testing the Bump Allocator

***What’s the Goal?***  
This task focuses on building and testing a **bump allocator**, which is a fast, memory-efficient custom allocator. The bump allocator provides fixed-size memory allocation for objects, resetting the memory when no longer needed.

***What’s New Here?***

* Built a custom bump allocator (BumpAllocator) that allocates memory in a fixed block.
* Provides functionality to allocate, deallocate, and reset the memory.
* Tests were written to confirm the behaviour of the allocator, including edge cases like exceeding memory capacity.

***How to Run It:***

1. Open the terminal in VS Code.
2. Compile and run the test\_allocator.cpp program:

g++ test\_allocator.cpp -o test\_allocator && ./test\_allocator

***What You’ll See:***  
The program runs multiple allocation and deallocation tests and prints the allocator's status. Example:

* Allocated memory for integers and structs.
* Successfully handles out-of-memory errors (e.g., exceeding capacity).
* Resets memory after deallocation.

**Task 2 Documentation Outline**

**Title**: Task 2 - Advanced Testing for the Bump Allocator

***What’s the Goal?***  
In this task, the allocator is tested more rigorously to ensure it handles various real-world scenarios. Tests include allocating multiple types, partial and full resets, and handling overflows gracefully.

***What’s New Here?***

* A series of tests were implemented to evaluate the allocator's functionality and robustness.
* Key scenarios tested:
  + Allocating memory for integers, doubles, and arrays.
  + Checking status after allocation.
  + Resetting the allocator partially and fully.
  + Exceeding capacity and ensuring it fails gracefully.

***How to Run It:***

1. Open the terminal in VS Code.
2. Compile and run the bump\_test.cpp program:

g++ bump\_test.cpp -o bump\_test && ./bump\_test

***What You’ll See:***  
The tests will output step-by-step results, such as:

* Allocator status after each operation (capacity, used memory, allocations).
* Messages confirming allocation success or failure.
* Results after partial and full resets.

**Task 3 Documentation Outline**

**Title**: Task 3 - Benchmarking the Bump Allocator

***What’s the Goal?***  
Task 3 benchmarks two versions of the bump allocator:

* **Bump Up Allocator**: Allocates memory upwards.
* **Bump Down Allocator**: Allocates memory downwards.

The goal is to compare their performance and assess which is more efficient under specific conditions.

***What’s New Here?***

* A benchmark library (benchmark.hpp) was implemented to measure execution time.
* Created tests to benchmark both allocation strategies:
  + Allocating memory upwards.
  + Allocating memory downwards.
* The benchmark measures the time taken for a large number of allocations and resets.

***How to Run It:***

1. Open the terminal in VS Code.
2. Compile and run the test\_bump\_allocator.cpp program:

g++ test\_bump\_allocator.cpp -o test\_bump\_allocator && ./test\_bump\_allocator

**What You’ll See:**  
The benchmark outputs the execution time for each allocation strategy:

Benchmark "Bump Up Allocator": <time in ms>

Benchmark "Bump Down Allocator": <time in ms>