use Adobe Reader to complete

(Type in fields)

PA3 – Memory

Student Information

Integrity Policy: All university integrity and class syllabus policies have been followed. I have neither given, nor received, nor have I tolerated others' use of unauthorized aid.

I understand and followed these policies: [✓ Yes] No

Name:	Joey Domino
Date:	5/15/2022

Submission Details

Final ***Changelist*** number:	
Verified build:	[✓ Yes] No
Number Tests Passed:	45
Required Configurations:	2: { Debug/Release | x86 }

Discussion (What did you learn):

Why is alignment such a pain in the ass all of a sudden?!

Verify Builds

- Follow the Piazza procedure on submission
  - Verify your submission compiles and works at the changelist number.
- Verify that only MINIMUM files are submitted
  - No – Generated files
    - *.pdb, *.suo, *.sdf, *.user, *.obj, *.exe, *.log, *.pdb, *.db, *.user
    - Anything that is generated by the compiler should not be included
  - No – Generated directories
    - /Debug, /Release, /Log, /ipch, /.vs
- Typical files project files that are required
  - *.sln, *.cpp, *.h
  - *.vcxproj, *.vcxproj.filters, CleanMe.bat

Standard Rules

Submit multiple times to Perforce

- Submit your work as you go to perforce several times (at least 5)
  - As soon as you get something working, submit to perforce
  - Have reasonable check-in comments
    - Points will be deducted if minimum is not reached

Write all programs in cross-platform C++

- Optimize for execution speed and robustness
- Working code doesn’t mean full credit

Submission Report

- Fill out the submission Report
  - No report, no grade

Code and project needs to compile and run

- Make sure that your program compiles and runs
  - Warning level ALL …
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
  - Code needs to work “as-is”.
    - No modifications to files or deleting files necessary to compile or run.
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Project needs to run to completion

- If it crashes for any reason…
  - It will not be graded and you get a 0

No Containers

- NO STL allowed {Vector, Lists, Sets, etc...}
  - Except for <Unorder Set> used in registration
  - No automatic containers or arrays
  - You need to do this the old fashion way - ***YOU EARNED IT***

Leave Project Settings

- Do NOT change the project or warning level
  - Any changing of level or suppression of warnings is an integrity issue

Simple C++

- No modern C++
  - No Lambdas, Autos, templates, etc…
  - No Boost
- NO Streams
  - Used fopen, fread, fwrite…
- No code in MACROS
  - Code needs to be in cpp files to see and debug it easy
- ***Exception:***
  - implicit problem needs templates

Leaking Memory

- If the program leaks memory
  - There is a deduction of 20% of grade
- If a class creates an object using new/malloc
  - It is responsible for its deletion
- Any ***MEMORY*** dynamically allocated that isn't freed up is ***LEAKING***
  - Leaking is ***HORRIBLE***, so you lose points

No Debug code or files disabled

- Make sure the program is returned to the original state
  - If you added debug code, please return to original state
- If you disabled file, you need to re-enable the files
  - All files must be active to get credit.
  - Better to lose points for unit tests than to disable and lose all points

Allowed to Add files to this project

- This project will work "as-is" do not add files…

UnitTestConfiguration file (if provided) needs to be set by user

- Grading will be on the UnitTestConfiguration settings
  - Please explicitly set which tests you want graded… no regrading if set incorrectly

Due Dates

- See Piazza for due date and time
- Submit program perforce in your student directory assignment supplied.
- Fill out your this ***Submission Report*** and commit to perforce
  - ***ONLY*** use Adobe Reader to fill out form, all others will be rejected.
  - Fill out the form and discussion for full credit.

Goals

- Create a Simple Memory system
  - Learn how to use Heaps, Memory Allocations, Debug Information
  - Learn how to wrap a system in to a library
  - Learn Win32 API

Assignments

1. **Create the Memory program in C++, for Visual Studio 2019**
   a. Document the code
   b. Code should be Warning Level Wall free
      1. It doesn't compile → 0 for grade
   c. Compiles in all 2 configurations
      - Debug x86
      - Release x86

2. **Program should be able to create and destroy a memory system.**
   a. Memory system should support multiple heaps
      1. Ability to create and destroy heaps
   b. Allocation from specific heaps
   c. Per Allocation track debug information
      1. File
      2. Line number
   d. Tracking blocks
      1. Heaps should also have the same tracking information as the other memory blocks of your existing memory system.
         1. Such as __FILE__, __LINE__
         2. Ability to dump the blocks on a specific heap

3. Heap tracking info
    a. Num of used blocks
    b. Peak num of used blocks
    c. Total size of heap
    d. Heap name

3. Memory system must wrap existing win32 memory systems

a. Use Win32 memory system calls like:
    1. HeapCreate(), HeapDestroy(), HeapAllocate(), HeapFree()
    2. HeapReAlloc(), HeapSize()

4. Add debug information to allocations

a. Overload the new and new[] operator to add more parameters
b. Feel free to add macros if desired to store even more implicit parameters like
    1. __FILE__
    2. __LINE__
c. You can restrict this to malloc and free allocations, but be careful to insure that all new/delete operators are being wrapped with your malloc and free operations.

5. Adding alignment

a. Your system should allow the ability to add alignment.
    1. Common alignments are:
        1. 4,8,16,32,64,128,256 alignments
        2. Enumerations are very useful to keep interfaces clean
b. You can have a default alignment setting if desired.
    1. Generally, that's 4-Byte aligned.

6. Debugging

a. Make sure you can walk and dump a list of every allocation
    1. Per Heap
b. Make sure you can walk and dump a list of every allocation
    1. Across the complete engine
c. Enough of debug info to fine an arbitrary memory leak.

7. General Testing

a. Make sure your memory system can support several heaps
b. Different allocations
c. Different alignments
d. Several debug functions to help you track blocks of memory

Validation

Simple checklist to make sure that everything is submitted correctly

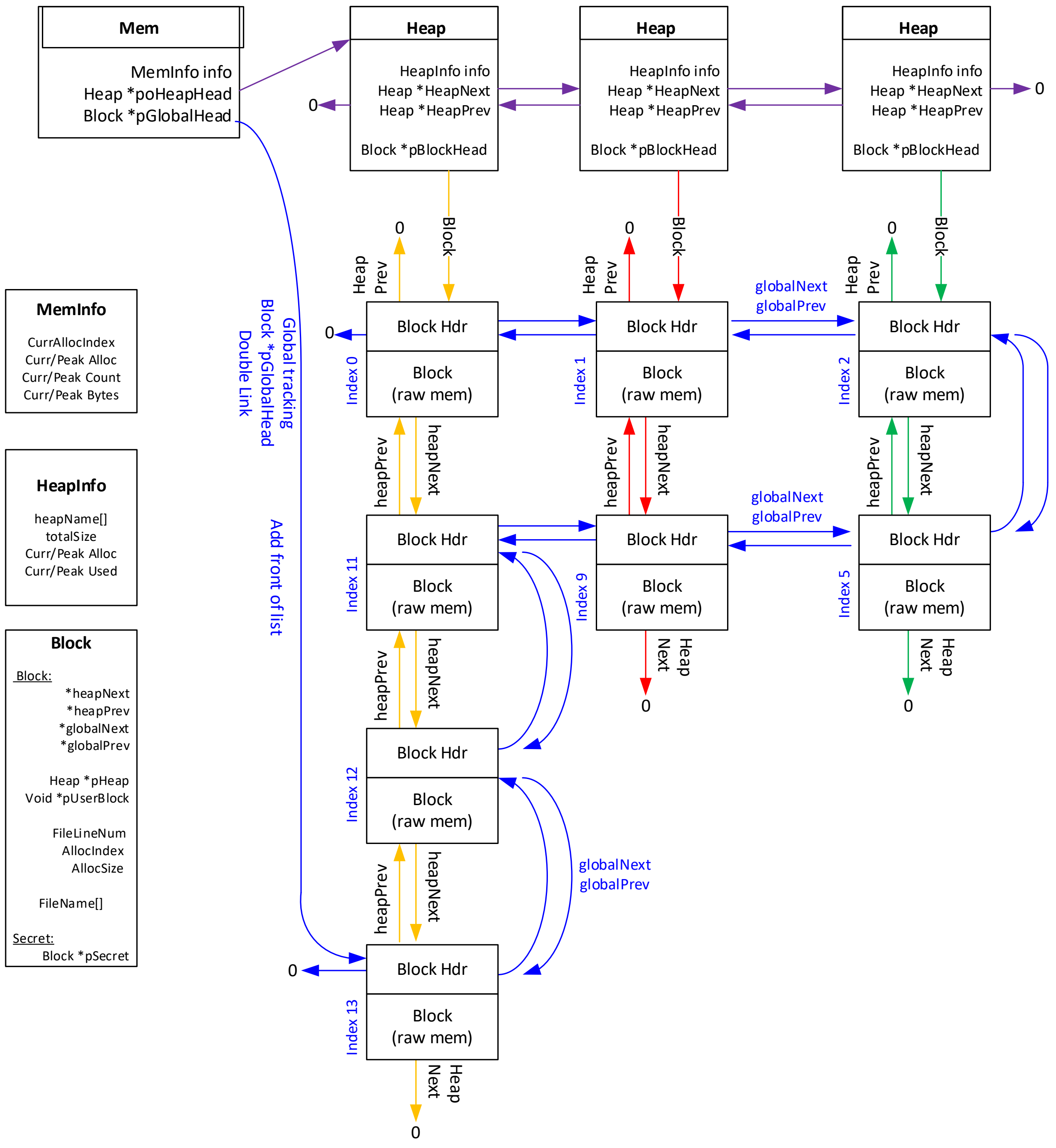
- Is the project compiling and running without any errors or warnings?
- Does the project run ***ALL*** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
- Follow the verification process for perforce
    - Is all the code there and compiles "as-is"?
    - No extra files
- Is the project leaking memory?

Hints

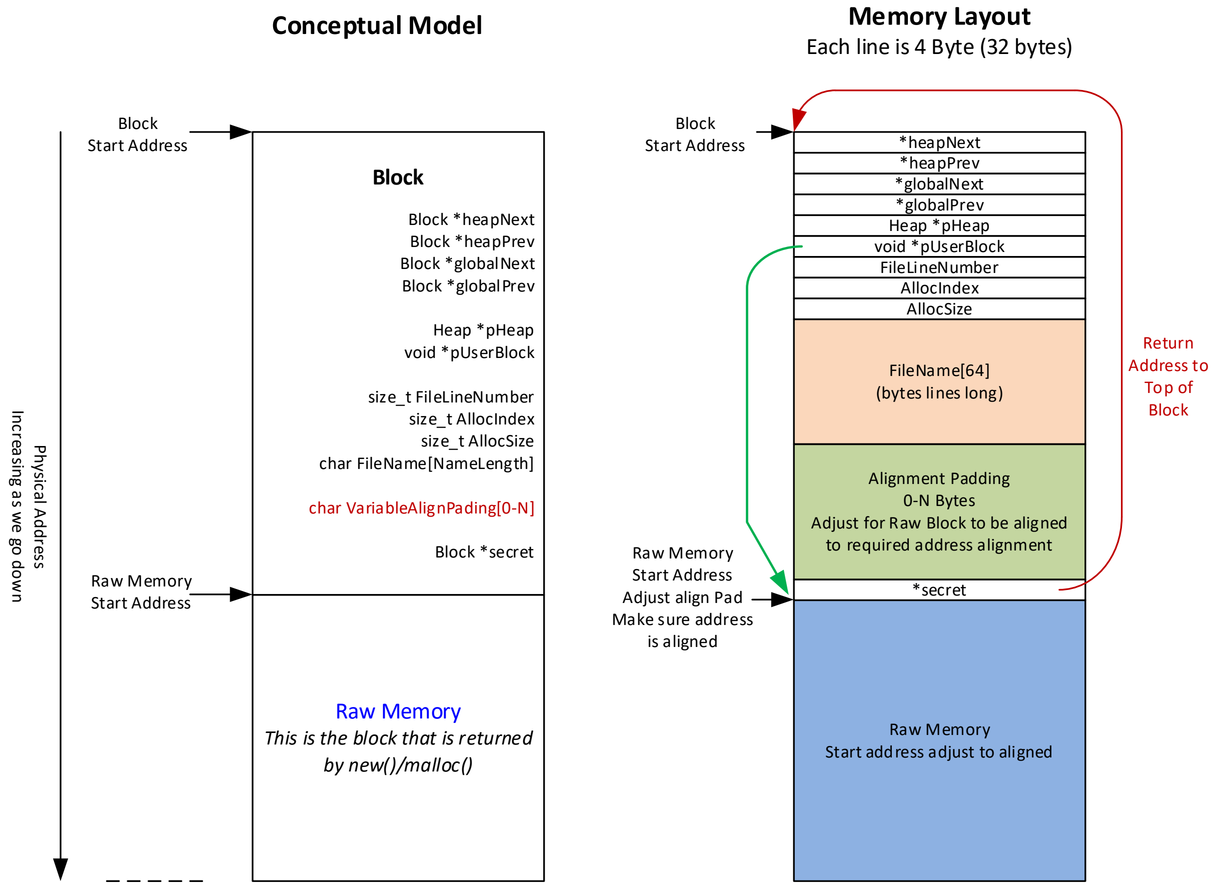
Most assignments will have hints in a section like this.

- Do this assignment by iterating and slowly growing your project
- Write your use cases first
- Write your API first and try to write your test code first.
- Don't be a cowboy or cowgirl
    - Write a wrapper on top of system calls like HeapAllocate()…
    - If you get this completely done, and you still want to…, then add a custom memory system.

Heap-Based Memory System



Block Layout


Conceptual Model
Memory Layout
Each line is 4 Byte (32 bytes)
Block Start Address
Block
Block *heapNext
Block *heapPrev
Block *globalNext
Block *globalPrev
Heap *pHeap
void *pUserBlock
size_t FileLineNumber
size_t AllocIndex
size_t AllocSize
char FileName[NameLength]
char VariableAlignPading[0-N]
Block *secret
Raw Memory Start Address
Raw Memory
This is the block that is returned by new()/malloc()
Physical Address
Increasing as we go down
Block Start Address
*heapNext
*heapPrev
*globalNext
*globalPrev
Heap *pHeap
void *pUserBlock
FileLineNumber
AllocIndex
AllocSize
FileName[64]
(bytes lines long)
Alignment Padding
0-N Bytes
Adjust for Raw Block to be aligned to required address alignment
*secret
Raw Memory Start Address Adjust align Pad Make sure address is aligned
Raw Memory
Start address adjust to aligned
Return Address to Top of Block