use Adobe Reader to complete

(Type in fields)

PA3 – Memory part B

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:	Thang Tran
Date:	10/6/2018

Submission Details

Final ***Changelist*** number:	317797
Verified build:	● Yes ○ No
Number Tests Passed:	13
Required Configurations:	DEBUG & RELEASE

Discussion (What did you learn):

I learned how to initialize, allocate, and free memory in a more advanced level as well as deal with edge cases.

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
    - 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, *.suo,
  - *.vcxproj, *.vcxproj.filters, *.vcxproj.user
  - *.cpp, *.h
  - 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...}
  - 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

No Adding files to this project

- This project will work "as-is" do not add files…
- Grading system will overwrite project settings and will ignore any student's added files and will returned program to the original state

use Adobe Reader to complete

(Type in fields)

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

- Learn
    - To Create a Memory System from scratch
- Understand the internals of a memory system

Assignments

1. Please ***VERIFY*** your builds for both ***DEBUG*** and ***RELEASE***

2. Create a memory system within a heap

- Take the given memory system framework for the heap layout:
    - Add the allocators
    - Add the de-allocators
- Run the Test functions that handles a set of memory allocation and de-allocations
    - *Supplied by Instructor*
    - ***Part B: tests 1-16 + stress test(17)***
        - Cut and paste your work from Part A into Part B
        - Continue developing
- Diagram the data structure layout out - to help you.

3. Take Memory system, use the stress test

- Measure the timing with default setting in the compiler
    - For the original memory system
    - For your custom memory system
- Measure the difference.
    - See in the Output windows
- Instructor will provide the stress tests
    - Make sure your program runs all unit tests 1-16
    - Make sure it runs the stress test – without crashing

4. *Grading*

a. Points:

- 17 Unit tests (last one is the stress test)
- 5 points for stress test
  - Working and performance time
- You cannot run the stress test UNLESS unit tests 1-17 are working.
  - 3 pts for Unit test 16
    - (Checking coalescing on adjacent free blocks - no looping)
- Points
  - 17 pts – unit tests
  - 3 pts – unit test 16 - adjacent free blocks… no looping
  - 5 pts – stress test
  - total: 25 pts

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

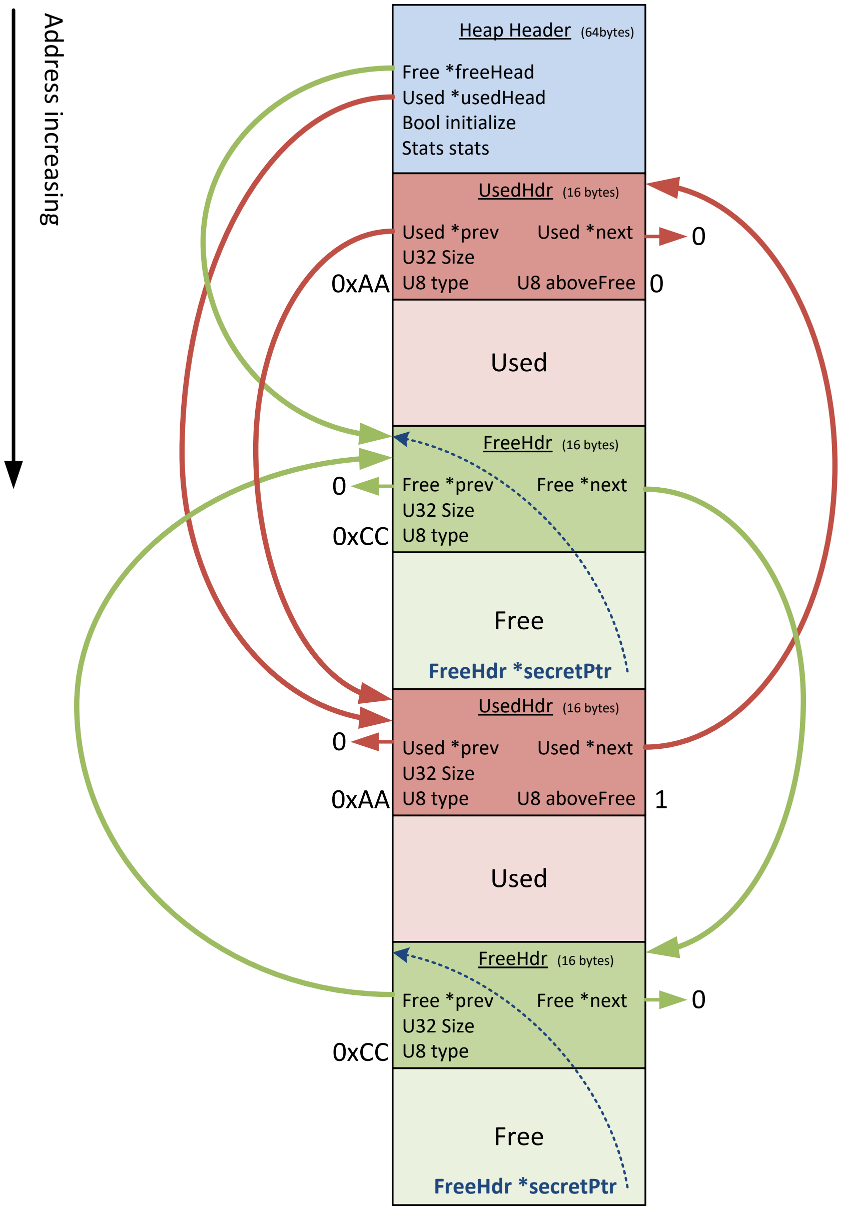
Most assignments will have hints in a section like this.

- Do many little check-ins
  - Iteration is easy and it helps.
  - Perforce is good at it.
- Look at the lecture notes!
  - A lot of good ideas in there.
  - The code in the examples work.
- Use the Piazza
  - This is much harder than the last assignment.
  - See me during office hours.
  - Read, explore, ask questions in class

use Adobe Reader to complete

(Type in fields)

Memory system diagram:



Notes:

* Used blocks are unsorted, pushed to the head

* Free blocks are sorted smallest address at the front of list

* Used block size, Free block size does not include the header size

* Minimum allocation is multiple of 16 bytes

* Heap is aligned on creation, no need to align the heap after it has been initialized

* Two adjacent free blocks are coalesced into one large free block

* secretPtr is place at the bottom 32 bits of the free block, it points back to the freeHdr

* types – 0xAA used, 0xCC free