PA7 – File System

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:	kaijun he
Date:	03/04/2019

Submission Details

Final ***Changelist*** number:	346592
Verified build:	[✓ Yes] No
Number Tests Passed:	3
Required Configurations:	Debug & Release
GRAD or UNDERGRAD:	GRAD

Discussion (What did you learn):

from this PA, i learn how to using loadinPlace tech to avoid using new to increase performance, and how to do double linked list using insertion and merge sorting method.

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...}
    - 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

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

- Learn
    - File Basics
        - fopen, fread, fclose, fseek,
        - (hopefully it's a review)
    - Load a dynamic memory in-place file for fast reload and run
    - Sorting linked lists
        - Merge, Insertion, combination techniques

Assignments

- Please ***VERIFY*** the correct builds for each project

Section 1: EVERYONE Sorting exercise

- Create 3 sorting routines for double linked lists
    - Insertion sort
        1. Use this as reference material,
            1. http://quiz.geeksforgeeks.org/insertion-sort-for-singly-linked-list/
            2. Port this code into your project
        2. Run the benchmark for timings
    - Merge sort
        1. Use this as reference material,
            1. http://www.geeksforgeeks.org/merge-sort-for-linked-list/
            2. Port this code into your project
        2. Run the benchmark for timings
    - Merge / Insertion combo sort
        1. Use the above sorts and create a hybrid sort
        2. Where the list is sorted with Merge, when the sub list get under a certain cutoff length, it switches into the insertion sort
    - Run the benchmark for timings

Section 2: <u>Under Grads only</u> - Basic file load and restore.

- Based on the linked list provided….
- Write code to copy the node data (many nodes) to a single binary file
    - Code must be in BINARY mode
        1. Only use fopen, fread, ….
        2. No Streams allowed
        3. No Boost STL
        4. No Modern C++
        5. Old school - BABY!
        6. Make sure its BINARY - not text mode
- Write code to load data from your binary file
    - Create methods to write and read data to the file
        1. You can add methods and data to OList class if you want
    - Recreate the linked list from your loaded data
- Run validation program

Section 3: <u>Grads only</u> – Write a Load in Place file (Contiguous memory footprint)

- Write Contiguous memory footprint to a new binary file
    - add any extra data necessary for pointer fix-up
- Load this data from a binary file into ONE memory block
    - perform pointer fix-up
    - reconstruction cannot exceed 1-3 new calls
- Run validation program

Section 4: <u>EVERYONE</u> – Do Not submit your binary file

- The file is auto-generated every time the program is executed

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 ***<u>ALL</u>*** the unit tests execute without crashing?
- Is the submission report filled in and submitted to perforce?
    - Fill out the form (make sure you specify GRAD or UNDERGRAD)
- 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.

- Practice your file system stuff
- Create several example solutions with different file patterns – fopen, fread, fwrite
- Make sure you are using the binary operation and not the text mode.
    - ‘wt’ – write text ← BAD
    - ‘wb’ – write binary ← GOOD
    - Same for read
- Look up file read / write examples from the internet or out of the book
    - I like the fopen, fwrite way of doing stuff as opposed to the streams.
    - Stay with old style for this assignment
- Use the FORUMs
    - This is much harder than the last assignment.
    - See me during office hours.
    - Read, explore, ask questions in class