PA5 – Lions and Tigers

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:

Date:

Submission Details

Final ***Changelist*** number:

Verified build: ◯ Yes ◯ No

Number Tests Passed:

Required Configurations: various configurations

Discussion (What did you learn):

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
  - Implicit, Return Value Opt, Proxy, Compiler settings
  - Understand C++ language from an optimization perspective

Assignments

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

- ***Implicit conversions***
  - Need to build in 3 configurations:
    - DEBUG
    - RELEASE
    - PREVENT
  - ***Debug*** configuration
    - Do not modify code or any compiler settings
    - It's just here as a timing reference
  - ***Release*** configuration
    - Do not modify code or any compiler settings
    - It's just here as a timing reference
  - ***PREVENT*** configuration ← DO your work here
    - Add code to ***Implicit.h*** / ***Implicit.cpp*** to prevent implicit conversions of data
    - This should prevent the code from compiling
    - It should generate errors
    - I will be grading on the number and types of errors generated

- ***Return Value Optimizations (RVO)***
  - Need to build in 2 configurations:
    - DEBUG
    - RELEASE
  - Open the ***Debug*** solution… Rework Files to add RVO ← DO your work here
    - Modify ***RVO.h*** and ***RVO.cpp*** to add Return Value Optimization
  - ***Debug*** configuration
    - Do not modify any compiler settings
  - ***Release*** configuration
    - Do not modify any compiler settings

- ***Proxy objects***
  - Need to build in 2 configurations:
    - DEBUG
    - RELEASE
  - Open the ***Debug*** solution… Rework Files to add Proxy ← DO your work here
    - Modify ***Proxy.h*** and ***Proxy.cpp*** to add proxy objects
    - Note:
      - Due to the improved optimization ability of the compiler
        - Some restrictions apply…
      - You cannot implement this function in the header, you should be implementing the appropriate proxy with 5 vectors.

```
Vect2D operator + (const Vect2D &tmp) const;
```

  - ***Debug*** configuration
    - Do not modify any compiler settings
  - ***Release*** configuration
    - Do not modify any compiler settings

- ***C++ Benchmarks*** – Compiler tweaking for speed
  - Need to build in 3 configurations:
    - DEBUG
    - RELEASE
    - MR_FAST
  - No modifications of Benchmark files
    - This part of the assignment is only for compiler tweaks
    - Do NOT change any line of code for this project
  - ***Debug*** configuration
    - Do not modify any compiler settings
  - ***Release*** configuration
    - Do not modify any compiler settings
  - ***MR_FAST*** configuration ← DO your work here
    - Research and adjust compiler settings
      - Try to improve the performance numbers
      - Look at the numbers from Debug and Release for comparison
    - Make all changes ONLY to MR_FAST configuration
      - Please be careful not to change Release or Debug only modify MR_FAST configuration
    - Record any modifications that improved speed in a text file
      - Fill out the ***Benchmark_MR_FAST_SETTINGS.txt*** with your modifications

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?
- Fill out the ***Benchmark_MR_FAST_SETTINGS.txt***
- 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!
  - Many good ideas in there.
  - The code in the examples work.
- Use the Piazza
- Proxies are the hardest section
  - Start on this early
  - Study the lecture inside out
  - Read the book on that section