PA5 – Lions, Tigers and Bears (C++ Efficiency)

Due Date

- See Piazza for due date and time
  - Grading the next day
- Submit program to perforce in your student directory
  - Sub directory called:
    - /PA5/...
  - Fill out your ***PA5 Submission Report.pdf***
    - Place it in the same directory as your solution
    - Enter the final Changelist number of your submission
    - Write up a quick discussion in the report
      - What you learned from this assignment so far

Goals

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

Assignments

RUN ALL TESTS in **VISUAL STUDIO** with **F5** (with debugging)

1. ***Convert a given Class A to remove implicit conversions***
   - Do Not Change ANY compiler settings - this project is very sensitive
     - Convert all methods to prevent the implicit conversion
   - Test the code Class A
     - Originally compiled with the unaltered class
       - Only warnings (no errors)
       - Run in Debug / Release mode
       - Save the text files - (need them later)
         - Debug - Implicit conversion.txt
         - Release - Implicit conversion.txt
     - Compile with the new protected class
       - Should generate compiler errors
       - Cut and paste the compiler errors into a txt file
         - Errors - for implicit conversion
         - Error.txt - (need the file later)

    - Switch the data to all floats
        - Run the code again in Debug / Release mode
        - Save the text files - need them later
            - Debug - NO Implicit conversion.txt
            - Release - NO Implicit conversion.txt
- Please create an Output file to mimic the reference text.
    - You can cut and paste this output file by hand
        - That's why you saved the previous 5 files
        - To use for cut and paste

2. Convert a given Class B to use Return Value Optimizations

- Do Not Change ANY compiler settings - this project is very sensitive
- Convert the class B to use the return value optimizations
    - Single step the code in debug and release
    - Print constructors and destructors to help prove to you that it's working
        - Only for your use, do not test with those prints
- Loop the code to show that there are savings with and without RVO
    - Used the supplied timer
    - Stress it and see the performance difference
    - Output the findings to a text file
- Please create an Output file to mimic the reference text.
    - You can cut and paste this output file by hand
    - Same as before
        - You need {With/Without RVO} for {Debug/Release}
        - So it will take 4 runs to get the complete data

3. Convert a given Class C rework to use proxy object

- Do Not Change ANY compiler settings - this project is very sensitive
- Convert the class C to use proxy object optimizations
- Write an more optimized function at the center of the proxy object
    - Loop the code to show that there are savings with and without proxy objects
        - Used the supplied timer
        - Stress it and see the performance difference
        - Output the findings to a text file
- Please create an Output file to mimic the reference text.
    - You can cut and paste this output file by hand
        - same as before

4. ***Take C++ Benchmarks as the stress test***
    - Measure the timing with default setting in the compiler
        - For the original C++ Benchmarks
    - Adjust and research the compiler settings
    - Measure the difference.
        - Write the before and after in an output text file
    - Instructor will provide the ***C++ Benchmarks*** program
        - ***Make all your changes in MR_FAST mode, not anything else***
    - Please create Output files to mimic the reference text.
        - You can cut and paste this output file by hand
        - Also list your modifications to the compiler in that MR_FAST file

General:

- Write all programs in cross-platform C or C++.
    - Optimize for execution speed and robustness.
- Create a programming file for each problem, for example
    - Student directory
        - /PA5/P5_Implicit/…
        - /PA5/ P5_Proxy/…
        - /PA5/ P5_RVO/…
        - /PA5/ P5_ Benchmarks/…
    - Make sure that each problem can be compiled and run through the checked in solution
- Do all your work by yourself
    - Feel free to talk with others about setup, version control, ideas
    - But do not copy your friend's code.
        - Please don't - I can tell with my difference tools
    - Feel free to share ideas
- Fill out the submission Report
- Check in the problems multiple times, at least 3 times per problem
    - Have reasonable check-in comments
    - Seriously, I'm checking
- Make sure that your program compiles and runs
    - Warning level 4, some times that is not possible due to MS headers…
    - Your code should be squeaky clean.

- We are using Perforce
  - You should have received the document describing how to login.
    - Please look at the documentation and videos under the reference directory
  - Submit program to perforce in your student directory
    - Sub directory called: /PA5/…
      - As described above
  - All your code must compile from perforce with no modifications.
    - Otherwise it's a 0, no exceptions

Validation

Simple check list to make sure that everything is checked in correctly

- Did you do all 4 problems?
- Do they compile and run without any errors?
- Warning level 4 free (or as close as you can go)?
- Submitted it into /PA5 directory?
- Can you delete you local drive, regrab the PA5 directory?
  - Is all the code there?
  - Does it compile?
- Did you check in your text files?

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 different than the last assignment.