PA4 – Lions, Tigers and Bears (C++ Efficiency)

Due Date

- See Piazza for any changes to due date and time
  - Friday by midnight
  - Grading the next day Saturday Morning
- Submit program to perforce in your student directory
  - Sub directory called:
    - /PA4/...
  - Fill out your ***PA4 Submission Report.pdf***
    - Use ADOBE READER
    - Enter the final Changelist number of your submission
    - Enter the number of problems completed
    - Write up a quick discussion in the report
      - What you learned from this assignment

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)
       - Record all the timings (4 different executions combos)

```
// You need to run this 4 times for timing with DETECT_IMPLICIT_CONVERSIONS 0

        // IMPLICIT_CONVERSIONS  0
        //     1) Run Debug
        //     2) Run Release
        // IMPLICIT_CONVERSIONS  1
        //     3) Run Debug
        //     4) Run Release
```

    - Compile with the new protected class
        - Add implicit protection for the Class A
            - Look at lecture notes
        - Compile in ***DEBUG*** mode with ***DETECT_IMPLICIT_CONVERSIONS*** 1
            - Set: ***IMPLICIT_CONVERSIONS*** 1
                - Should generate compiler errors
                - Record errors (cut and paste)
            - Set: ***IMPLICIT_CONVERSIONS*** 0
                - Should NOT generate compiler errors
        - Repeat in ***RELEASE*** mode with ***DETECT_IMPLICIT_CONVERSIONS*** 1
            - Set: ***IMPLICIT_CONVERSIONS*** 1
                - Should generate compiler errors
                - Record errors (cut and paste)
            - Set: ***IMPLICIT_CONVERSIONS*** 0
                - Should NOT generate compiler errors
- 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 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
        - Record all the timings (4 different executions combos)

```
// You need to run this 4 times for timing

// RVO_ENABLE  0
//     1) Run Debug
//     2) Run Release
// RVO_ENABLE  1
//     3) Run Debug
//     4) Run Release
```

- Please create an Output file to mimic the reference text.
    - You can cut and paste this output file by hand
        - same as before

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 a 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
        - Record all the timings (4 different executions combos)

```
// You need to run this 4 times for timing

// PROXY_ENABLE  0
//     1) Run Debug
//     2) Run Release
// PROXY_ENABLE  1
//     3) Run Debug
//     4) Run Release
```

- 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
        - /PA4/P4_Implicit/…
        - /PA4/ P4_Proxy/…
        - /PA4/ P4_RVO/…
        - /PA4/ P4_ 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 ideas on Piazza
  - You are 100% responsible for all code
  - See syllabus about collaboration rules
- Fill out the submission Report

- 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
    - Seriously, I'm checking
- Make sure that your program compiles and runs
  - Warning level ALL sometimes that is not possible due to MS headers...
    - There are corrections around windows headers
  - NO Warnings or ERRORS
    - Your code should be squeaky clean.
- NO STL allowed {Vector, Lists, Sets, etc...}
  - No automatic containers or arrays
  - You need to do this the old fashion way - ***YOU EARNED IT***
- No modern C++
  - Lambdas, Autos, templates, etc...
  - Except implicit problem needs templates

Validation

Simple checklist 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 Wall free (or as close as you can go)?
- Submitted it into /PA4 directory?
- Can you delete your local drive, regrab the PA4 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.