Assignment 3 - Google Test

Re-submit Assignment

Due Saturday by 11:59pm **Points** 39 **Submitting** a file upload
File Types hpp, cpp, txt, and in **Available** Sep 22 at 12am - Sep 30 at 11:59pm 9 days

The purpose of this assignment is to give you an introduction to using Google Test, or gtest, to create and perform unit tests against your code. Additionally, you'll gain more experience with CMake to both integrate gtest and utilize multiple C++ files in a single project.

This assignment builds on the previous assignment by having you reorganize your random distribution code into a pair of .hpp and .cpp files, along with adding in some pre-defined gtest until testing code.

Assignment

Working from the previous assignment...

1. Correct any mistakes and/or missing pieces from Assignment 2.
2. Create a header file named `distributions.hpp` and place the the `DistributionPair` class and ***only*** the four function prototypes specified in Assignment 2 into this file.
3. Create an implementation file named `distributions.cpp`. Place the implementations for the four specified functions from Assignment 2, along with any other functions you may have written, into this file.
4. Create a file called `main.cpp` that contains only the main function and the code segment provided in Assignment 2 in the section entitled Sample Run.
5. Create an executable target in your CMakeLists.txt file named `RandDistributions` that utilizes all of the above code to create an application that generates and plots the random distributions.
6. Add Google Test to your CMakeLists.txt file, naming the testing target `UnitTestRunner`.
7. Use the provided file **TestDistributions.cpp** as the entry point for the unit test target. This file contains the necessary main entry point, along with all of the unit testing code.

When complete, your CMakeLists.txt file will have two executable targets, `RandDistributions` and `UnitTestRunner`. When CMake is run, it will generate a single project that generates both executables. A sample run of the `UnitTestRunner` is shown below.

Unit Test Code

The grader for the class (Cameron) revised the testing code from the previous assignment to work with gtest. The file is named **TestDistributions.cpp** and can be found through this **link**. You must integrate this code into your project as a gtest unit testing executable. Use the name `UnitTestRunner` as the `add_executable` target name for the gtest testing executable.

Submission Notes

- Turn in the following files
  - **main.cpp**
  - **distributions.hpp**, **distributions.cpp**
  - **TestDistributions.cpp** (provided for you)
  - **CMakeLists.txt, CMakeLists.txt.in**.
- CMakeLists.txt targets
  - Use ***EXACTLY*** `RandDistributions` as the name for the `add_executable` target for the application.
  - Use ***EXACTLY*** `UnitTestRunner` as the name for the `add_executable` target for the unit tests.
- Your code must compile without any warnings or compiler errors. See syllabus regarding code that has compiler errors.
- Your code must adhere to the CS 3460 coding standard: **link**
- Your code must be formatted through the use of the following clang-format configuration file: **link**

Sample Run - Unit Test Runner

The following shows the results of running the `UnitTestRunner` target, passing all of the unit tests.

```
[==========] Running 6 tests from 3 test suites.
[----------] Global test environment set-up.
[----------] 2 tests from UniformDistribution
[ RUN      ] UniformDistribution.ReturnsExpectedBins
[       OK ] UniformDistribution.ReturnsExpectedBins (17 ms)
[ RUN      ] UniformDistribution.HasCorrectTotalAcrossAllBins
[       OK ] UniformDistribution.HasCorrectTotalAcrossAllBins (15 ms)
[----------] 2 tests from UniformDistribution (33 ms total)

[----------] 2 tests from NormalDistribution
[ RUN      ] NormalDistribution.ReturnsExpectedBins
[       OK ] NormalDistribution.ReturnsExpectedBins (1 ms)
[ RUN      ] NormalDistribution.HasCorrectTotalAcrossAllBins
[       OK ] NormalDistribution.HasCorrectTotalAcrossAllBins (20 ms)
[----------] 2 tests from NormalDistribution (22 ms total)

[----------] 2 tests from PoissonDistribution
[ RUN      ] PoissonDistribution.ReturnsExpectedBins
[       OK ] PoissonDistribution.ReturnsExpectedBins (1 ms)
[ RUN      ] PoissonDistribution.HasCorrectTotalAcrossAllBins
[       OK ] PoissonDistribution.HasCorrectTotalAcrossAllBins (81 ms)
[----------] 2 tests from PoissonDistribution (84 ms total)

[----------] Global test environment tear-down
[==========] 6 tests from 3 test suites ran. (140 ms total)
[  PASSED  ] 6 tests.
```

Assignment 3 - Google Test

Criteria	Ratings		Pts
Compiles without any warnings (MSVC and g++). on Linux (-Wall -Wextra -pedantic) on Windows (/W4 /permissive-)	**2.0 pts Full Marks**	**0.0 pts No Marks**	2.0 pts
Follows required course code style. Must also be formatted using clang-format.	**2.0 pts Full Marks**	**0.0 pts No Marks**	2.0 pts
Passes all unit tests	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
Correct generateX and plotDistribution function implementations.	**10.0 pts Full Marks**	**0.0 pts No Marks**	10.0 pts
Correct distributions.hpp file.	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
Correct distributions.cpp file	**5.0 pts Full Marks**	**0.0 pts No Marks**	5.0 pts
Correct Google Test integration This covers getting CMakeLists.txt updated with Google Test and creating both executable targets, one for the application, one for the unit tests.	**10.0 pts Full Marks**	**0.0 pts No Marks**	10.0 pts
			Total Points: 39.0