Assignment 1b: "Histogram"

Given is a C++ program that computes a histogram for 30000000 randomly generated numbers (sample size). The generated values range from 0 to 9 and the number of bins is set to 10 so that the number of occurrences of each value is counted separately.

Requirements summary:

- Use **C++ Multithreading** to develop 2 parallel versions of the Histogram program (histogram.hpp). The first version should use C++ atomic operations and/or mutexes (histogram-atomic-mutex.cpp) and the second version should be your best effort implementation (histogram-best.cpp) that minimizes synchronization as much as possible.
- The resulting histogram program must always match the "**total**" output number from the serial implementation.
- The best-effort histogram should achieve a speedup of around 15.

Data Handling and Generation:

- Histogram data is stored in a one-dimensional vector.
- Generator struct generates random numbers in a given range.
- All structs are implemented with all fields and methods public for ease of usage and testing on the online platform.

Detailed Description

1. Histogram with Atomic/Mutex [implement in histogram-atmoic-mutex.hpp]

The main function should **create a num_threads threads**, all of which call the worker function:

```
void worker(int sample_count, histogram& h, int num_bins)
```

While this function needs to process the given work, the main function must make sure that spawned threads have completed, and the work has been allocated properly. Any potential data races and synchronization should be handled with C++ mutexes or atomic operations directly in the **histogram** struct.

2. Histogram Best Effort [implement in histogram histogram-best.hpp]

Implement the same functionality as the first version but avoid synchronization as much as possible. This version of the histogram should achieve speedup of around 15 on an ALMA cluster node.

3. Parsing input [already implemented in helper.hpp]

Your program should accept the following three parameters:

- **--num-threads** – an integer representing the desired number of threads
- **--num-bins** – an integer representing the desired number of bins
- **--sample-size** – an integer representing the desired sample size
- **--print-level** – an integer representing the amount of information program prints to the output

4. Expected Behavior

Assuming your executable is called “histogram”, your code will be executed with the following input parameters:

```
./histogram --num-threads x, where x is 1, 2, 4, 8, 16 or 32
```

The expected output is the following:

```
Bins: 10, sample size: 30000000, threads: 1
0:3001319
1:2998561
2:3002571
3:2998314
4:3000574
5:3000892
6:2997455
7:3001024
8:3001479
9:2997811
total:30000000
0.819826
```

Your program should not produce any warnings, no additional info about bad input parameters, or other print outs. The output and test results should always produce the correct “total” number on the online platform, while executing with multiple threads. The last line represents the execution time of the program.

To compile at home use:

- `g++ -o histogram -std=c++20 -lpthread -O2 histogram.cpp`

To compile on ALMA use:

- `/opt/global/gcc-11.2.0/bin/g++ -o histogram -std=c++20 -lpthread -O2 histogram.cpp`

To run on ALMA use (example):

- `srun --nodes=1 ./histogram --num-threads 32`

*replace histogram.cpp with other version of the code (e.g., histogram-atomic-mutex.cpp, histogram-best.cpp).

5. Submission Guidelines

The program must be submitted before the deadline on the online platform after it has passed all checks. Additionally, you need to run your code on ALMA, measure speedup Further information is provided in the lectures, tutorials and on Moodle.

The required speedup on ALMA is ~15* and you should measure speedup for 2, 4, 8, 16 and 32 threads (not on the frontend, but by using **srun** as shown above). The sequential execution time on Alma is around **0.86** seconds. Once you have the results, enter them on the online platform (the speedup.graph entry) and click the “save” button below the graph. Make sure that both the source code files, and the speedup graph are present on the online platform. The online platform should not be used for speedup measurements.

*It is ok if you only one configuration achieves a speedup close to this one, e.g., when using all available threads on ALMA.