CSE 687 Object Oriented Design Project

MapReduce

Phase #2

# Background

In Stage 2, you will be writing a standalone command-line program that is capable of running a user specified map and reduce function as specified in DLLs. The program must run as a single process, no threads or processes (yet). The program will take as input:

1. Directory where input files are stored
2. Directory for intermediary files (this can be defaulted, and a user provided one overrides the default).
3. Path for output file (this can be defaulted, and a user provided one overrides the default).
4. A path to a DLL that specifies the use-case specific:
   1. map function
   2. reduce function

DLLs should be loaded explicitly as details will not be known at compilation time!

# Methodology Requirements

1. You must work in a team of 2. If there is an odd number of students, there will be a single team of 3. No one will be allowed to work independently.
2. You must use C++ in conjunction with MS Visual Studio and GitHub (student account).
3. You may use anything in the std library and additionally may use anything in the BOOST (https://www.boost.org/) library.
4. All changes must go through a code review by your partner(s).
5. All partners must submit a reasonably equal number of check-ins for each project, as evidenced by git submissions.
6. All partners must treat each other with respect.

# Technical Requirements

1. Mapper/Reducer should be generalized using some combination of templates, inheritance, and polymorphism.
2. You should create a project within your solution that only contains the interfaces for the mapper and reducer. Both the DLLs and framework may have dependencies on this interface project. The MR framework and the use-case DLL’s should have no dependencies between them.
3. Code must have unit tests above 95% line coverage. You may use, Boost unit testing, Google unit tests, or your own unit testing.
4. Code formatting & comments should follow Google style guide: <https://google.github.io/styleguide/cppguide.html>
5. Errors, warnings, information, fatal errors, etc., will be logged. Boost Logging can be used (<https://www.boost.org/doc/libs/1_63_0/libs/log/doc/html/index.html>).

# Rubric

1. Code behavioral correctness: 50%

Along with above requirements, code must reasonably handle all error scenarios. Code must never crash or hang.

1. Code Unit testing: 10%

Code must have code coverage above 95% by line count.

1. Code Commenting & Style: 10%

Code must meet style & comment guidelines.

1. Code Organization and Structure: 10%

Code must adhere to SOLID principles.

1. Procedural Correctness: 10%

Code must have gone through code reviews. Code reviews should be sufficiently thorough. Members that do not give their team quality code-reviews will be penalized.

Additionally, each member of the team must contribute a reasonably equal amount to their project. Each member will submit a brief report detailing what percentage of work was completed by each member. I encourage teams to keep detailed meeting notes.

# Bonus

You may only attempt the bonus if the code is behaviorally correct, unit tested, correctly formatted, and reasonably structured.

10%: Accept a 3rd DLL from the user for input formatting. The Input format will specify what is a record (key and value). This record will be accepted by the Mapper. For raw text files, there should be an input formatter that returns a line at a time as the value from one of the input files.