G2.1 **Testing and Refining Sines.** (Group project)

This week, each group should use our C++ template to translating your final algorithm for evaluating \( \sin(x) \) into a C++ code that the testers can evaluate.

For the next two weeks, each group should choose a portion of the evaluation process, and be prepared for the systematic study of everyone’s algorithm as the C++ codes become available. Tasks to be done include:

(a) **Speed testers:**
- Create example template (C++?) for fast-sine function
- Create timing class, evaluating at \( 10^6 \) random points, timing only evaluations
- Optimize compilation options

(b) **Error testers:** Using example template, set up
- Graphical error estimate
- Systematic searches for maximum error (for polynomial and rational function fits)
- Contemplate the challenge of finding maximum errors for splines and trapezoidal approximations.
- For methods that feasibly can produce more-than-machine accuracy, look to the High Precision group to provide appropriate routines

(c) **Machine performance group**
- Find machine timing information for adds, multiplies, divides, integer, mod, other operations
- Note complexities with more than one FPU, operation per cycle
- Note complexities involving cache
- Evaluate your fast-sine method: are you reaching near-ideal performance?
- For methods that feasibly can produce more-than-machine accuracy, look to the High Precision group to provide appropriate routines

(d) **High Precision group (for methods at greater than machine precision)**
- Generate classes for high-precision arithmetic (or use NR)
- Generate template high-precision-sine function for users
- Distribute both prototype and arithmetic classes

(e) **Pareto efficiency group**
- Plotting accuracy versus time for your method and system sine, and be prepared to incorporate data for other methods
- Find Pareto frontier