CPSC 2720 – Assignment 2

## Overview

In this assignment, you will:

* Write implementation for the various geometric shapes that were tested in the first assignment.
* Keep track of your progress using version control.
* Run code coverage to ensure all code is tested.
* Document your implementation using doxygen.
* Check for memory leaks using valgrind.

### Instructions

1. Fork the repository at <http://ares-mat17.cs.uleth.ca/gitlab/cpsc2720/Geometry/asn2>. As it is a CS department server, you will only be able to do this on the campus network (or via VPN).
2. Set your notification settings for this repository to “Watch” so you will receive email notification if there are any changes to repository (e.g. clarifications are added to the instructions).
3. Fork the repository so you have your own copy.
4. Set the project visibility for your forked repository to “Private”.
5. Add the marker as a member of your project with the permission “Reporter”. You will be provided with their CS department user name in the lab and/or on Moodle. This is needed so the marker can grade your assignment.
6. Setup your GitLab repository for running continuous integration for your project.
   1. Set the *Git Strategy* to “git clone”
   2. Set the Timeout to 5 (i.e. 5 minutes). Your CI job will be small, so this should be lots of time and will prevent any infinite loops from tying up the CI server

#### Completing the Assignment

1. Create a local clone of your assignment repository.
   1. Run the command git remote and verify that there is a remote called origin.
      1. origin is the link to your repository of GitLab and is where you will be pushing your changes.
2. Compile and run the provided code. It is suggested you do this using Code::Blocks by creating a console project and adding the files from the repository. A Makefile is also provided, if you prefer to use that.  
     
   If you have problems with the Code::Blocks project, check the following settings:
   1. The cppUnit library is linked in.
      1. Open the *Build options* for the project.
      2. Go to the *Linker settings* tab.
      3. Type –lcppunit in the *Other linker options* textbox
   2. Code::Blocks knows where to find the files.
      1. Open the *Build options* for the project.
      2. Go to the *Search directories* tab
      3. Check that the *Compiler* tab has the include directory where the header files are.
      4. Make sure that the directory name (e.g. ../asn2) matches the name of the directory for your assignment.
3. Generate the project documentation using doxygen or look at comments in the header (.h) files to see the specification of the methods.
4. Read through the generated documentation for all of the methods in all of the classes to understand the expected output of the methods.
5. Copy your unit test files from Assignment #1 into the test directory.
6. **Write implementation for the methods of the concrete classes** (e.g. Quadrilateral, Cone, Circle). Calculated values are considered equal to the nearest to 1/1000th (i.e. 0.12345 and 0.12389 are considered equal.).
7. You can use the formula abs(a-b) < 0.001 where a and b are doubles to check if two doubles are “close enough”.
8. You may find the following pages useful, as they contain geometric formulas:
9. <http://www.math-salamanders.com/image-files/geometry-terms-and-definitions-geometry-cheat-sheet-4-2d-shapes-formulas.gif>
10. <http://www.math-salamanders.com/image-files/high-school-geometry-help-geometry-cheat-sheet-5-3d-shape-formulas.gif>
11. Use your unit tests from Assignment #1 to verify your implementation.
    1. All the unit tests should fail initially, as there is no implementation (yet) for the methods.
    2. It is recommended that you use a Test-Driven Development to completing the assignment. This will help you to focus on implementing one shape class at a time.
    3. Add one unit test file (e.g. TestQuad.cpp) to the Code::Blocks project at a time
    4. Get all the tests to pass.
    5. Continue adding test fixtures and implement that class until all of the classes are completed and all tests pass.
12. A Makefile is provided that allows you to run code coverage to see if your unit tests are covering all of your source code. Based on the results, you may need to update your unit tests.
    1. To run the code coverage tool lcov, use the command:  
         
        make coverage  
         
       This will create the directory coverage which contains HTML files showing how much of your code your unit tests exercised.
    2. Open the file index.html in a web browser.
    3. You are expected to have 90% or better coverage for the concrete classes.
13. To run valgrind to check for memory leaks, use the command:  
      
     make memcheck  
      
    This will create an XML file showing the output of valgrind.
    1. Open the file memcheck.xml.
    2. The output is expected to show “0 blocks lost” in the summary.

# Grading

You will be graded based on your demonstrated understanding of the use of version control and good software engineering design practices. Examples of items the grader will be looking for include (but are not limited to):

* All methods of all concrete classes are tested by unit tests.
* Version control history shows an iterative approach to completing the assignment.
* Implementation shows understanding of software engineering design principles.
* Source code is appropriately documented using the principles discussed in class (e.g. all classes and methods) so that doxygen can extract the documentation.
* valgrind shows that no memory blocks were lost in the implementation.

### Submission

Submit the URL of your forked repository to Moodle. The grader will clone your repository as of the deadline for grading.

# Appendices

## Updating the Assignment Files

The following information is to be used in the case that the assignment is updated with clarifications or corrections.

1. Create an upstream remote so you can pull in the updates:  
     
   git remote add upstream <http://ares-mat17.cs.uleth.ca/gitlab/cpsc2720/Geometry/asn2.git>  
     
   This command creates a link to the assignment repository. You will not have permissions to push to it, so you will get an error if you try.
2. To get updates from the assignment repository, you can pull them into your local repository.  
     
   git pull upstream master