**CSE 687 Object Oriented Design Project**

**Test Harness**

**Phase #1 – Architecture and Design**

***Overall Purpose:***

Developing software consisting of multiple packages requires frequent testing. If the software has complex features, it is best to build it incrementally. First design and implement a very basic core with a small number of packages, then add features one-at-a-time by adding new packages, or adding a few lines of code to an existing package. Each time new functionality is added, developers should build and test. That way, if additions break existing code, we know where to look, e.g., in the newly added few lines of code. A test harness allows developers to use that incremental process efficiently.

A test harness should allow developers to define and aggregate many small tests, each of which runs with exception handling and results logging. The goal is to do that without littering code with many try-catch blocks and logging statements. The technique used to do that is to define, in the test harness, an execution function that accepts a callable test object (FN #1) - function pointer, functor, or lambda - and invokes the object within the scope of a try block. A catch clause displays exceptions and returns false to indicate test failure. Otherwise the executor returns the test predicate returned by the test object, e.g., true or false for pass or fail.

The test executor also provides results logging with predefined messages. A test harness should define several levels of logging, e.g., just pass or fail, or more detailed, test-specific messages, or very detailed debugging messages, which display a time-date stamp and the values of key test-defined variables.

The test harness also provides a mechanism to link any number of tests into a test sequence. Typically, during development, tests are added for each new feature, and they can be re-run just for that feature, or all the tests of all the features, to ensure that the new addition didn't break existing functionality.

***Phases:***

You will be developing this project in four phases:

1. Architecture & Design
2. Test Harness Development
3. Multiple Test Capability
4. Remote Operation Capability

Each phase will move your project closer to completion, and you will receive a set of instructions for each phase. The instructions will include any new requirements (additional functionality) for the phase.

***Phase #1:***

In phase #1, you are to create the architecture and basic design of your test harness.

**Architecture**

The architecture of your project describes the various components that will make up your Test Harness. These are not the actual classes and data you will be developing, but more of a functional description.

The architecture of your project must include a system block diagram which shows each high-level component of your Test Harness system, along with any major sub-components. The block diagram must also include interface definitions – connections between the high-level components, and connections between the major sub-components within a high-level component.

This architecture (and related material) may change during the development of your project and will be submitted as part of each Phase.

**Design**

The design of your project is intended to show how you will break down the architectural functionality into actual classes, including data. You can describe your design in class diagrams (UML), pictures, text descriptions, or any combination of these. You must indicate how the design relates to the architecture – which classes implement which component(s), and what data are flowing through each interface.

As with the architecture, your design may also change throughout the development process, and will need to be updated and submitted at each step.

Additionally, I would like each team to describe how each team-mate is contributing to the overall success of the group. This should be done in a high-level way. For instance, you may say “The team worked together through Zoom to collaborate on each portion of the Phase #1 development.” Or you may indicate “J was the main architect, B led the design effort, and D was in charge of interfaces. The entire team was responsible for overall review.”

***Overall Requirements:***

**Project Requirements**

1. The test system **shall** be written in the standard C++ programming language, using the standard C++ libraries.

**Test Harness Requirements**

1. The test system **shall** provide a Test Harness which accepts any callable object that requires no arguments (FN #2) and returns a boolean predicate to indicate success or failure.
2. The Test Harness **shall**  provide a multi-level logging mechanism, intended to show:
   * just pass-fail status
   * application specific messages for pass and fail, along with the result
   * detailed debugging output that includes level-two logging and a time-date stamp
3. The Test Harness **shall** invoke a passed callable object in the scope of a try block.
4. If exceptions are thrown during execution of the test, the catch block **shall** log test failure and (for level two or level three logging) the contents of the exception message.

**Testing Requirements**

1. Test Drivers **shall** implement a common interface that declares a test method taking no arguments and returning a boolean value indicating test pass or failure.
2. The test system **shall** provide a sequence of tests demonstrating all the features of your TestHarness.

***Submission:***

All files associated with your Project Phase #1 will be submitted by a single team member. Additionally, the team will present the material to the professor and answer questions.

***Foot Notes:***

1. A callable object is an object that can be invoked as if it were a function. These will be described in detail in several of the lectures.
2. You can provide arguments for the execution as member data of a functor, or as data captured by a lambda.