Examination #1

Name:_____ **Instructor's Solution** _____ SUID:_____

This is a closed book examination. Please place all your books on the floor beside you. You may keep one page of notes on your desktop in addition to this exam package. All examinations will be collected promptly at the end of the class period. Please be prepared to quickly hand in your examination at that time.

If you have any questions, please do not leave your seat. Raise your hand and I will come to your desk to discuss your question. I will answer all questions about the meaning of the wording of any question. I may choose not to answer other questions.

You will find it helpful to review all questions before beginning. All questions are given equal weight for grading, but not all questions have the same difficulty. Therefore, it is very much to your advantage to answer first those questions you believe to be easiest.

1. While writing C# code, when would you choose to use a delegate. Please describe all the kinds of uses you can think of. Note that this question does not need and should not include application detail.

   Answer:

   Delegates are a means to define processing that is deferred until some later time.

   Delegates are used to publish events to registered subscribers. Frameworks like WinForms and WPF define delegates for events that applications will subscribe to, like button clicks.

   They also are used to bind to lambdas in order to transport them from one scope to another. This is what happens when you use the Dispatcher to pass an action to be performed from a child thread to a UI thread. It is also what happens when you define a lambda as the argument of a Thread constructor.

   They also support asynchronous calls using BeginInvoke and EndInvoke.

   In Summary:

   - Publish and subscribe for event handling
   - Bind to a lambda for creating a thread or passing to a UI thread
   - Pass to a function to use for callback when function's processing completes
   - Asynchronous calls without explicitly creating a thread or task

2. Assume that your TestHarness class design, for Project #4, defines a method:

   TestResults runTests(TestRequest tr) { … }

   Which does all the things you are required to do when processing a TestRequest message. Show how you would run this method concurrently when there are several TestRequest messages enqueued in a BlockingQueue<Message>. Show how you would limit the maximum number of threads used for this purpose.

   Answer:

```
  ThreadStart doTests = () => {

    while(true)
    {
      Message testRequest = inQ_.deQ();

      if(testRequest.body == "quit")
      {
        inQ_.enQ(testRequest);
        return;
      }
      ITestResults testResults = runTests(testRequest);

      lock (sync_)
      {
        client_.sendResults(makeTestResultsMessage(testResults));
      }
    }
  };
  int numThreads = 8;
  for(int i = 0; i < numThreads; ++i)
  {
    Console.Write("\n  Creating AppDomain thread");
    Thread t = new Thread(doTests);
    threads_.Add(t);
    t.Start();
  }
}

// Pseudo Code:
// ============
// 1. Define lambda that, in a forever loop, deQs message
//    and invokes testRequest, then sends results to client
//    and returns to deQ another message.
// 2. Create and start the desired number of threads using a
//    ThreadStart delegate bound to the lambda.
```

3. Discuss all the important Critical Issues for Project #4, just as you should in an Operational Concept Document (OCD).

   Answer:

   - **Issue: Demonstrate requirements effectively** in asynchronous environment
     Solution:
     Use console with client GUI, TestHarness, and Repository. Write to console (or log) for each action that implements a requirement, tagged with req. # and time stamp.
   - **Issue: Concurrent access to files in Repository** may cause stream opening conflicts
     Solution:
     Use retry wrapper (See my sol to MT2Q1) and report error if max retries are exceeded.
   - **Issue**: **GUI design may be complex** due to number of messages that have to be processed
     Solution:
     Use tabbed display with tabs for connecting and for each type of message sent. Each message tab has a message creator method that the tab calls (to keep Window code simple).
   - **Issue**: **Graceful shutdown of multiple processing threads** in TestHarness
     Solution:
     Have each thread re-enqueue quit message before exiting so all threads will shutdown. Will need to keep a list of active threads to wait on after sending quit message.
   - **Issue**: **May have large testing loads**
     Solution:
     Can add additional TestHarness Servers, as each TestRequest is independent of other testing. Will need to keep a list of all running servers and their endpoints that can be read by all clients and Repository so clients can find a newly added server.
   - **Issue**: **Testing the Tester**
     Automated testing of asynchronous systems like TestHarness can be difficult, since the outputs are not always generated in the same order, e.g., they are less predictable than for synchronous systems.
     Solution:
     Develop a set of test messages that are rich enough to exercise all of the System's functionality. Log responses at every step of processing in a list of responses. With a validated system or by analyzing the TH code, record the responses and store for later comparisons. When comparing just check to see if a response is in the list and, at the end, every list response has been checked.
   - **Issue**: **Ease of Use** - how do we make system easy for users to accomplish their work tasks?
     Solution: Use agents to allow uses to build macro like capability to implement their tasks. Perhaps we provide a standard agent class that knows how to traverse the Repository and have it accept a lambda that defines some user defined processing. It would be interesting if we could use some Python script to define the processing.

   Note:
   It isn't terribly important that your issues exactly match mine. What is important is that they are important, are not simply design issues that can be resolved by any competent developer, and that you have stated a coherent way to resolve the issue.

4. Write all the code for Service and Data contracts for Project #4.

Answer:

```
[ServiceContract(Namespace = "MTF16")]
public interface IStreamService
{
  [OperationContract(IsOneWay = true)]
  void upLoadFile(FileTransferMessage msg);
  [OperationContract]
  Stream downLoadFile(string filename);
}
[MessageContract]
public class FileTransferMessage
{
  [MessageHeader(MustUnderstand = true)]
  public string filename { get; set; }

  [MessageBodyMember(Order = 1)]
  public Stream transferStream { get; set; }
}
[ServiceContract(Namespace="MTF16")]
public interface IMessageService
{
  [OperationContract(IsOneWay = true)]
  void PostMessage(Message msg);

  Message GetMessage();
}
[DataContract]
public class Message
{
  [DataMember]
  public string type { get; set; } = "default";
  [DataMember]
  public string author { get; set; } = "Fawcett";
  [DataMember]
  public string to { get; set; } = "http://localhost:8080/ICommService";
  [DataMember]
  public string from { get; set; } = "http://localhost:8081/ICommService";
  [DataMember]
  public DateTime timeStamp { get; set; } = DateTime.Now;
  [DataMember]
  public string body { get; set; } = "";
}
```

If you use File chunking as in FileService-SelfHost, then you can put both file handling and message handling in the same service, since they both can use WSHttpBinding. I've used two services here because using streams requires BasicHttpBinding and I want to use WSHttpBinding for message handling to preserve message order.

5. What is meant by the term Continuous Integration?

   Answer:

   I've encouraged you to implement your projects by developing a concept and then build the project's parts by adding a few lines of code to a working code base, build, and test, to ensure that the changes are correct, keep the system working, and don't break any other functionality.

   Continuous Integration is a process we use for large systems that is a scaling out of the process described in the previous paragraph. We use tools like Test Harnesses, Repositories, and Clients to make small modifications and/or additions to the current baseline, rebuild, and test. We need tools like the TestHarness and Repository to automate testing, because any one test may be run hundreds of times as the code it depends on changes.

6. Describe the syntax and semantics of passing a C# function a single reference argument by reference.

   Answer:

   The syntax requires use of the ref qualifier for the function parameter(s) for both definition and invocation, e.g.:

   Definition:

```
void myFunction(ref ReferenceType myReferenceType) { … }
```

   Invocation:

```
ReferenceType myReferenceType = new ReferenceType();
myFunction(ref myReferenceType);
```

   If myFunction changes the state of the myReferenceType argument, the caller sees the change. If myFunction changes the object referred to, by assigning it a new instance of ReferenceType, the caller sees the new instance.

7. Draw an activity diagram for your design of Project #4's Client.

Answer: