CSE686 Midterm #1

Name: ____Instructor's Solution_____ SUID: _____

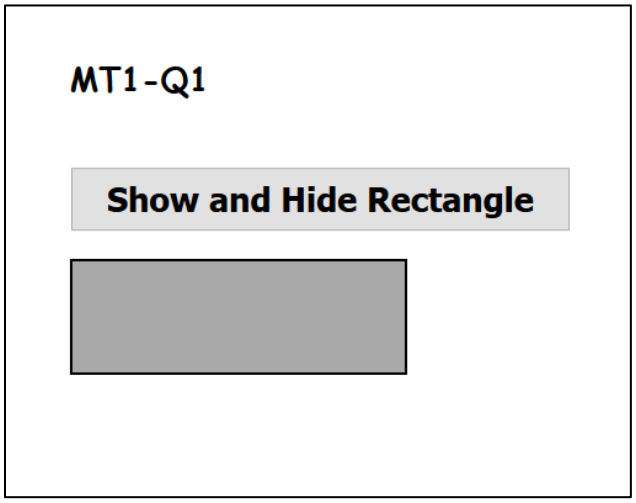
This is a closed book examination. Please place all your books on the floor beside you. You may keep one page of notes on your desktop in addition to this exam package. All Exams will be collected promptly at the end of the class period. Please be prepared to quickly hand in your examination at that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

1. Write all the code and markup to display a webpage with a button and immediately below, a rectangle with gray fill. The button is required to toggle the rectangle between visible state (default) and not visible.

```
<style>
  body {
    padding: 1em 2em;
    font-family: 'Comic Sans MS';
  }
  #TheButton {
    font-size:large; font-weight:bold;
    padding: 0.25em 1em;
    margin-top: 1em;
  }
  #TheRectangle {
    border: 2px solid black;
    background-color: darkgray;
    margin-top: 1em;
    height: 4em;
    width: 12em;
    display: block;
  }
</style>
<script>
  function toggleVisibility() {
    var elem = document.getElementById("TheRectangle");
    if (elem.style.display !== "none")
      elem.style.display = "none";
    else
      elem.style.display = "block";
  }
</script>
</head>
<body>
  <h3>MT1-Q1</h3>
  <button id="TheButton" onclick="toggleVisibility()">Show and Hide Rectangle</button>
  <div id="TheRectangle"></div>
</body>
```



2. Describe the Asp.Net Core pipeline, e.g., what is it and how does it work. You may write code to aid your explanation but are not required to do so.

   The Asp.Net Core pipeline is a chain of delegate processing elements, called middleware, which act on incoming requests, and may, as well, act on the returning response, in sequence. This sequence is configured in the

   `Startup.Configure(IApplicationBuilder app, IHostingEnvironment env, …)`

   Each middleware component is added with the app.Use(…) method or app.UseXXX extension methods. The IApplicationBuilder interface declares a method:

   `use(Func<RequestDelegate, RequestDelegate>)`

   That accepts a RequestDelegate (configured with a lambda in its component) and returns a Task (to the next middleware component). That delegate's first argument is the HttpContext, which has properties that hold the HttpRequest and HttpResponse.

   The framework provides several useXXX extension methods that help with specific middleware tasks. The framework also provides an extension method:

   `run(this IApplicationBuilder, RequestDelegate)`

   That terminates the pipeline.

   Each of the pipeline processing components may pass on the request to the next component in the pipeline sequence, or, may simply return a response to the requestor.

   Services used in pipeline processing components are registered for dependency injection in the method:

   `Startup.ConfigureServices(IServiceCollection services)`

   and injected into **`Startup.Configure`** method as an argument (the … in **`Startup.Configure`**), to be used by the **`app.UseXXX`** methods.

   Here's a link with very clear explanation of the pipeline operation:
   https://www.thomaslevesque.com/2018/03/27/understanding-the-asp-net-core-middleware-pipeline/

3. What is an MVC route? What does it do? Write code to declare a route, as you might do in your final project. Give three examples of browser addresses that are routed without error, where each address results in a different response.

An MVC route is a specification of where a request is to be handled, e.g., the Controller, the Action, and optionally, an index of an item in one of the MVC models. It looks like this:

```
app.UseMvc(routes =>
{
  routes.MapRoute(
    name: "default",
    template: "{controller=Home}/{action=Index}/{id?}");[1]
});
```

This statement is found in the Startup.Configure(…) method.

Here are three examples:

http://ipadress:port
request sent to default controller's default action (usually Index)

http://ipdress:port/Course
request sent to CourseController's default action

http://ipadress:port/Course/Edit/1
request sent to CourseController's Edit action with argument 1.

[1] There is an overload that accepts key url and expects an object value, e.g., new { controller="aContr", … }