|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Components of the MVC pattern** ➢ The **model** consists of the code that provides the data access and business logic. ➢ The **view** consists of the code that generates the user interface and presents it to the user. ➢ The **controller** consists of the code that receives requests from users, gets the appropriate data from the model, and passes that data to the appropriate view. | | | Starting up VS  ASP.NET MVC C#--> ASP.NET Web Application (.NET Framework)  MVC | |
| **Structure of a MVC project**  • **App\_Data**: Data files, such as database files or XML. • **App\_Start:** Classes that are executed on starts up. • **Content**: Static resources like CSS and images. • **Controllers**: Controller classes (HTTP requests, process input, output). • **fonts**: Font files (special or default). • **Models**: Classes that represent the data and business logic. • **Scripts**: JavaScript files. • **Views**: View templates that display the user interface. • **bin**: Contains compiled code and assemblies (hidden). • **obj**: Temporary files generated during build (hidden). • **Properties**: Application settings and metadata. • **packages**: Third-party libraries and dependencies (separate folder). | | **ADDING a CLASS (Model)**  Model 🡪 Add 🡪 Class 🡪 “Class” + name i.e MyClass.cs  **ADDING a CONTROLLER (Reason: Adding a new controller to enable HTTP request from the user UI to be processed and a result to be returned)**  Controllers 🡪 Add 🡪 Controller 🡪 Empty 🡪 “NameController.cs”  **ADDING a VIEW**  Views 🡪 Home 🡪 Add 🡪 View 🡪 “Name.cshtml” | | |
| **1. Model**  - Handles **data & business logic**.  - Interacts with the **database**.  Example:  public class Person {  public int Id { get; set; }  public string Name { get; set; }  } | **2. View**  - Defines the **UI using Razor syntax**.  - Displays data from Model.  - Sends user input to the Controller for processing.  - Receives updates from the Controller  Example:  @model List<Person>  @foreach(var person in Model) {  <p>@person.Name</p>  } | | | **3. Controller**  - Handles **HTTP requests**.  - Calls Model and passes data to View.  - User actions trigger the Controller, which updates the Model if necessary, and the Model then usually notifies the View that a change has occurred  - Example:  public class HomeController : Controller {  public ActionResult Index() {  var people = new List<Person>{ new Person { Name = "John" } };  return View(people);  }  } |

**Model View Controller**

**CSS Box Model**

|  |  |
| --- | --- |
| **The CSS Box Model** is a fundamental concept that defines how elements are structured and displayed on a web page.  Every element is treated as a rectangular box consisting of multiple layers that affect its overall size and spacing.  Each HTML element is wrapped by a box.  Everything in CSS has a box around it, and understanding these boxes is key to being able to create layouts with CSS, or to align items with other items.  • Content  The content of the box, where text and images appear.  • Padding  Clears an area around the content. The padding is transparent  • Border  A border that goes around the padding and content  • Margin  Clears an area outside the border. The margin is transparent. Default is 0. It is important to note that margin is the external space separating boxes | |
| **Calculation of the dimension of the box** p { width: 200px; padding: 10px; margin: 10px; border: 1px solid #eee; } **Total width**: width+ padding-left + padding-right + border-right + border-left = 222px  **Total Height** = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin | **Box-Sizing Property:**  **content-box (default):** The width and height only include the content.  Padding and border are added *outside* the specified width and height, increasing the total element size.  **padding-box:** The width and height include padding but *not* the border.  **border-box:** The width and height include content, padding, and border.  This makes layout calculations more predictable and is commonly used (\* { box-sizing: border-box; } ensures all elements follow this rule). |
| **Why border-box is useful:**  • It prevents the total size of an element from exceeding its specified dimensions. • It ensures a more intuitive design, especially for grid-based layouts.  • By default, browsers use border-box for <table>, <select>, <button>, and <input> elements of type radio, checkbox, reset, button, submit, color, and search. |
| **Handling Overflow**  **visible (default):** Content overflows without being clipped.  **hidden:** Extra content is clipped and not visible.  **scroll: A** scrollbar appears even if the content fits within the element.  **auto:** A scrollbar appears only when the content overflows. | **Creating Shapes in CSS:**   1. Use nested div elements for the dartboard: outer (green), middle (yellow), and inner (red). 2. Apply border-radius: 50% to make circles, ensuring equal width and height. 3. Centre elements using display: flex; align-items: center; justify-content: center;. 4. Set different background colors (darkgreen, goldenrod, darkred) and border styles. 5. To create an oval, set width larger than height and use border-radius: 50%. 6. Example CSS:. oval { width: 200px; height: 100px; border-radius: 50%; }. 7. Follow same steps as circle for oval |

**CSS Display Property CSS Position Property**

|  |  |
| --- | --- |
| The display property defines how elements are rendered in the layout.  **Inline Elements (display: inline):**  Do not break onto a new line.  • Only take up as much width as necessary.  • Width and height properties are ignored.  • Examples: <a>, <span>, <strong>, <em>.  **Inline Replaced Elements:**  • Have an intrinsic width and height set outside CSS.  • Content is usually external (e.g., images, inputs).  • Examples: <img>, <input>.  **Block Elements (display: block):**  • Always start on a new line.  • Take up the full width of the parent container.  • Width, height, padding, and margins are respected.  • Examples: <p>, <div>, <h1>, <form>, <table>.  **Inline-Block Elements (display: inline-block)**  • Behaves like inline, but allows setting width and height.  • Top and bottom margins/padding are respected.  • Elements appear next to each other without line breaks | Determines how an element is positioned relative to the document flow. Possible values:  **Static (default):**  • Elements are positioned according to the normal document flow.  • top, left, right, bottom values have no effect.  **Relative:**  • Elements remain in the normal document flow.  • Can be offset using top, left, right, bottom (relative to its original position).  **Absolute:**  • The element is removed from the normal document flow.  • Positioned relative to the closest positioned ancestor (not static).  • If no ancestor is positioned, it is relative to <html>.  **Fixed:**  • Like absolute, but always positioned relative to <html>.  • The element does not move when scrolling. |

**Advanced CSS Positioning concepts**

|  |  |  |
| --- | --- | --- |
| **1. Z-Index:**  • z-index controls the **stacking order** of elements.  • Higher values place elements **in front**, while lower values place elements **behind** others.  • Only works on elements with position: relative, absolute, or fixed (not static).  Example:  .box1 {  position: absolute;  z-index: 2; /\* Will appear in front \*/ }  .box2 {  position: absolute;  z-index: 1; /\* Will appear behind \*/}  A z-index of -1 places the element behind other elements. | **2. Float Property:**  The float property is used to position elements within their container,  often for wrapping text around images.  **Common Values:**  • left: Element floats to the left of its container.  • right: Element floats to the right of its container.  • none: Default; element remains in normal flow.  • inherit: Inherits the float value from its parent.  Example:  img {  float: left;  margin-right: 10px;}  This will make text wrap around an image positioned on the left. | **Clearing Floats**  When using float, other elements may be affected. Use clear to prevent layout issues:  .clearfix::after {  content: "";  display: block;  clear: both;}  Apply .clearfix to the parent container to fix float-related issues. |

**Pseudo Classes**

|  |  |  |
| --- | --- | --- |
| Pseudo classes apply styles based on an element’s state.  Common Pseudo classes include: (not included:, such as :not(), :checked, and :nth-of-type(). ) | | |
| **User Interaction:**  :hover → Styles an element when hovered.  :focus → Styles an element when it receives focus.  :active → Styles an element when clicked.  Example:  button:hover {  background-color: blue;}  input:focus {  border: 2px solid red;} | **Link Styling:**  :link → Styles unvisited links.  :visited → Styles visited links.  Example:  a:link {  color: blue;}  a:visited {  color: purple;} | **Structural Pseudo-Classes:**  :first-child → Styles the first child of a parent.  :last-child → Styles the last child of a parent.  :nth-child(n) → Styles the nth child.  Example:  p:first-child {  color: red;}  p:nth-child(2) {  font-weight: bold;} |

**Cascading Style Sheets (CSS)**

|  |  |  |
| --- | --- | --- |
| is a style sheet language used to control the presentation of web documents written in HTML.  It is one of the core technologies of the World Wide Web alongside HTML and JavaScript.  CSS is designed to separate content from design enabling more flexible and maintainable web development | | |
| **• Internal CSS**\*\* - Internal styles are applied within a `<style>` tag inside an HTML document.  <style> body {padding-top: 70px;padding-bottom: 20px;}</style> | • **External CSS**\*\*- External styles are stored in a separate `.css` file and linked to the HTML document using a `<link>` tag.  <link rel="stylesheet"href="/css/main.css"> | •**Inline CSS**\*\*- Inline styles are applied directly to an individual HTML element using the `style` attribute.  <p style="height:182px !important;">This is a paragraph.</p> |

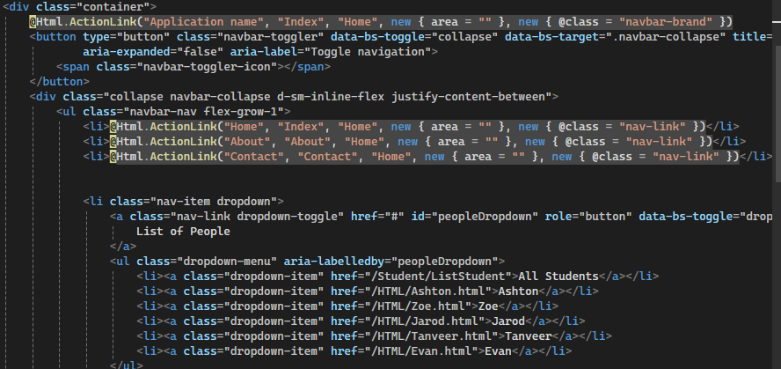
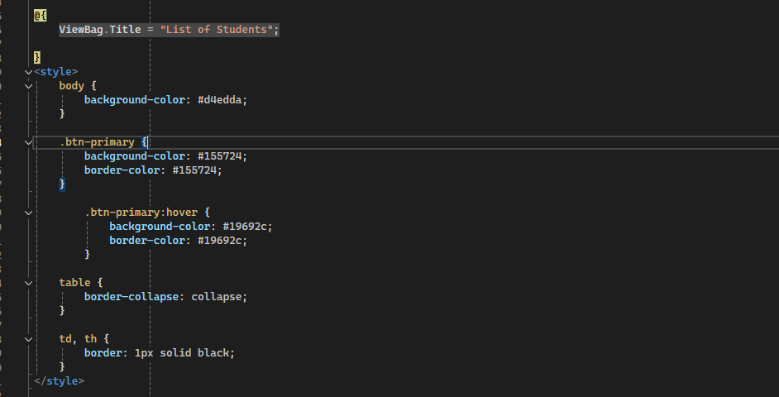
**Responsive Web Design (RWD)**

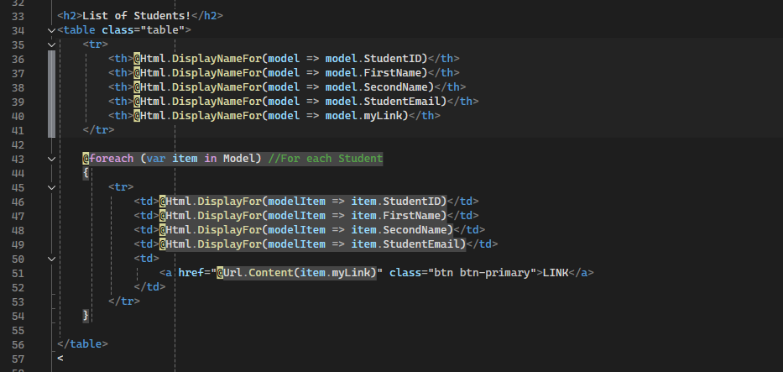
|  |  |  |
| --- | --- | --- |
| Responsive Web Design ensures that web pages are displayed properly on a variety of devices from desktop computers to smartphones. This is achieved through the use of CSS techniques such as media queries. | | |
| **Media Queries:** The @media rule allows different styles to be applied based on the type of device or its screen properties.  Example: @media (max-width: 1100px) {  body {  background-color: lightgrey;}  }  Media queries help deliver a tailored style sheet depending on:  •Screen size (mobile, tablet, desktop)  •Print styles for printed documents  •Accessibility features like screen readers | **Media Features:** Media features provide detailed conditions for applying styles based on specific device properties:  Example: @media (min-width: 600px) and (max-width: 1200px) {  body {  font-size: 18px;}}  This ensures that the font size is adjusted for screens within the specified range. | **Responsive Frameworks:** Responsive frameworks provide pre-built CSS and JavaScript components that make it easier to develop mobile-friendly websites.  **Popular Responsive Frameworks:**  •Bootstrap •Skeleton  •Foundation •Milligram  •Bulma •Spectre•Primer  •UIkit  •Semantic UI  •Materialize  •Tailwind CSS |

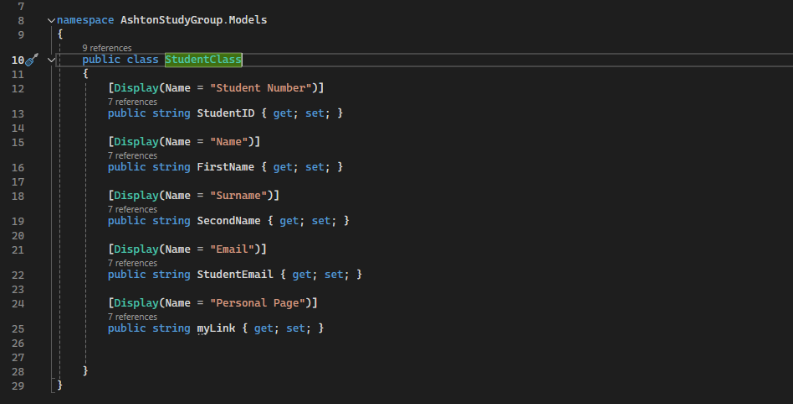
**BootStrap**

|  |  |
| --- | --- |
| Bootstrap is a widely used frontend framework that includes pre-styled components, layout grids, and JavaScript plugins to facilitate responsive web design. | |
| **Features of Bootstrap**  **HTML and CSS templates** for typography, forms, buttons, tables, navigation, modals, and image carousels.  **JavaScript plugins** for interactivity.  **Grid system** for flexible layouts.  **Mobile-first design**, ensuring content scales properly across all devices. | **Advantages of Bootstrap:**  **Easy to use**: Beginners with basic HTML and CSS knowledge can start using it immediately.  **Responsive design**: Ensures a consistent look and feel across different devices.  **Mobile-first approach**: The framework prioritizes mobile-friendly designs by default.  **Cross-browser compatibility**: Works with all modern web browsers. |

**Prac 2 Code:**

1. **Layout.cshtml 2. ListStudent.cshtml(View)**



1. **StudentClass (Model)**

4 **StudentController.cs (Control)+**

public ActionResult ListStudent()

{

List<Models.StudentClass> students = new List<Models.StudentClass>();

students.Add(new Models.StudentClass { FirstName = "Ashton", SecondName = "Theron", StudentEmail = "u24650677@tuks.co.za", StudentID = "u24650677", myLink = "~/HTML/Ashton.html" });

students.Add(new Models.StudentClass { FirstName = "Jarod", SecondName = "Chengan", StudentEmail = "u24445314@tuks.co.za", StudentID = "u24445314", myLink = "~/HTML/Jarod.html" });

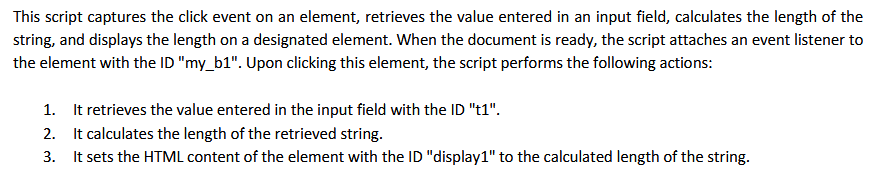
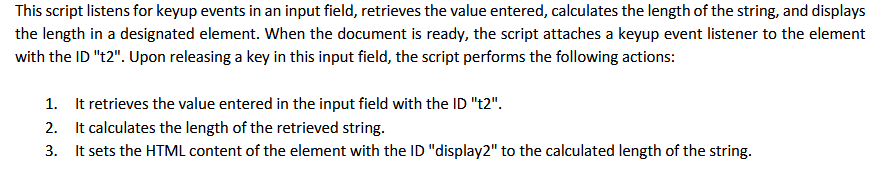
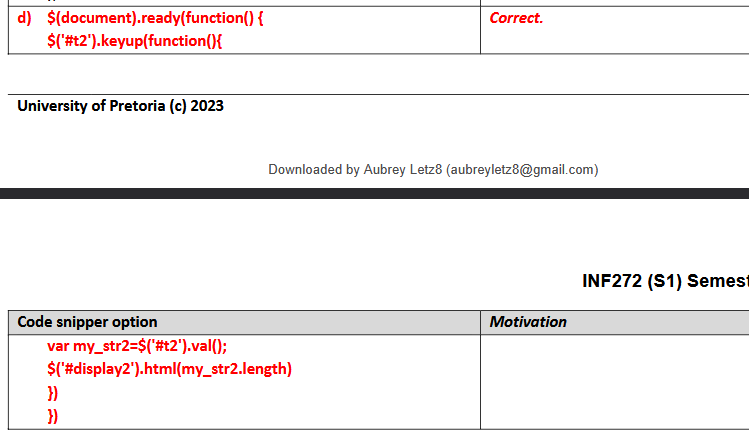
students.Add(new Models.StudentClass { FirstName = "Zoe", SecondName = "Joubert", StudentEmail = "u05326343@tuks.co.za", StudentID = "u05326343", myLink = "~/HTML/Zoe.html" });

students.Add(new Models.StudentClass { FirstName = "Evan", SecondName = "Barnes", StudentEmail = "u23573982@tuks.co.za", StudentID = "u23573982", myLink = "~/HTML/Evan.html" });

students.Add(new Models.StudentClass { FirstName = "Tanveer", SecondName = "Balmacont", StudentEmail = "u24047568@tuks.co.za", StudentID = "u24047568", myLink = "~/HTML/Tanveer.html" });

return View(students);

}



|  |  |
| --- | --- |
|  |  |

1. **HTML example (PRAC 2) CSS Example**

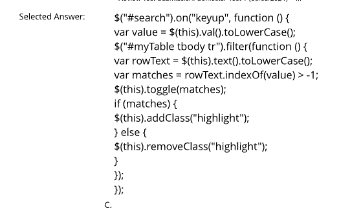
|  |  |  |
| --- | --- | --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Ashton - Study Group</title>  <link rel="stylesheet" href="styles.css">  </head>  <script src="script.js"></script>  <body div class="content">    <header>  <h1>Ashton</h1>  </header>  <nav>  <ul>  <li><a href="/Home/Index">Home</a></li>  <li><a href="/Student/ListStudent">List</a></li>  <li><a class="active" href="Ashton.html">Ashton</a></li>  <li><a href="Zoe.html">Zoe</a></li>  <li><a href="Jarod.html">Jarod</a></li>  <li><a href="Tanveer.html">Tanveer</a></li>  <li><a href="Evan.html">Evan</a></li>  <li><a href="#"onclick="goBack()">BACK</a></li>  </ul>  </nav>  <section class="bio-section">  <h2>Background</h2>  <div class="content-wrapper left">  <img src="Images/Nerd1.jpg" alt="Picture of Ashton">  <p>  So i come from Pretoria Boys high school and i typically do magic the gathering  followed by other card games and video games as well as coding  </p>  </div>  </section>  <section class="hobbies-section">  <h2>Hobbies</h2>  <div class="content-wrapper right">  <p>  I play a lot of magic the gathering and i am typically good at it, i also consider my Job  at the tuks gaming society as a hobby  </p>  <img src="Images/Job1.jpg" alt="Picture of Hobby">  </div>  </section>    <section class="modules-section">  <h2>Modules</h2>  <table>  <tr><th>Semester</th><th>Module</th></tr>  <tr><td>1</td><td>JCP202</td></tr>  <tr><td>1</td><td>FIL251</td></tr>  <tr><td>1</td><td>INF214</td></tr>  <tr><td>2</td><td>INF225</td></tr>  <tr><td>2</td><td>INF261</td></tr>  <tr><td>1</td><td>INF271</td></tr>  <tr><td>1</td><td>INF272</td></tr>  <tr><td>1</td><td>BAC200</td></tr>  <tr><td>1</td><td>IAU200</td></tr>  </table>  </section>  <footer>  <p>&copy; 2025 Study Group</p>  </footer>  </body>  </html> | \*  {  margin: 0;  padding: 0;  box-sizing: border-box;  }  html, body  {  background-color: #155724;  margin: 0;  padding: 0;  font-family: Arial, sans-serif;  }  ul  {  list-style-type: none;  margin: 0;  padding: 0;  display: flex;  background-color: #333;  flex-wrap: wrap;  }  li  {  flex: 0;  text-align: center;  }  li a  {  display: block;  color: white;  padding: 14px 16px;  text-decoration: none;  }  ul li:not(:last-child) {  border-right: 1px solid #fff; /\* Add border-right to all but the last item \*/  }  ul li:last-child {  flex-grow: 1; /\* so it grows all the way to the end\*/  text-align: left;  }  .active  {  background-color: #04AA6D;  }  li a.active  {  background-color: #04AA6D;  }  li a:hover  {  background-color: #111;  }  li a.active:hover  {  background-color: #04AA6D; | }  .content  {  max-width: 1000px;  margin: auto;  padding: 10px;  background: #d4edda;  border: 2px solid #28a745;  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);  }  img  {  height: auto;  display: block;  margin: auto;  max-width: 400px;  max-height: 400px;  }  section {  display: flex;  flex-direction: column;  padding: 20px;  border: 2px solid #28a745;  margin-bottom: 20px;  }  section img  {  width: 200px;  height: 200px;  object-fit: cover;  border-radius: 10px;  }  section img:hover  {  transform: scaleX(-1);  }  .content-wrapper  {  display: flex;  align-items: center;  gap: 20px;  }  .content-wrapper.left img  {  order: -1;  }  .content-wrapper.right img {  order: 1;  }  table th  {  text-align: center;  }  table td  {  text-align: center;  } |
| <script>  function validateForm() {  let isValid = true;  const firstName = document.getElementById('firstName');  const lastName = document.getElementById('lastName');  // validate First Name  if (firstName.value.trim() === '') {  firstName.classList.add('is-invalid');  isValid = false;  } else {  firstName.classList.remove('is-invalid');  } | // validate Last Name  if (lastName.value.trim() === '') {  lastName.classList.add('is-invalid');  isValid = false;  } else {  lastName.classList.remove('is-invalid');  }  // Email does not need to be validated via javascript, due to type="email"  return isValid;  }  </script> | |

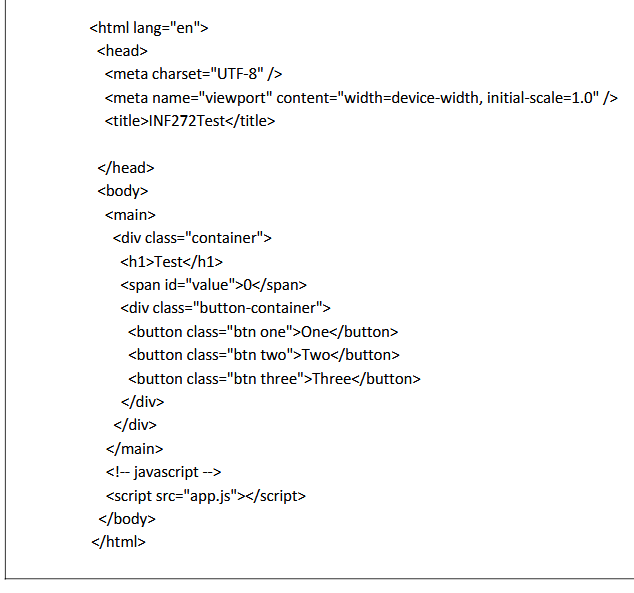
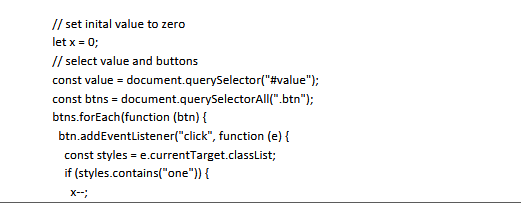
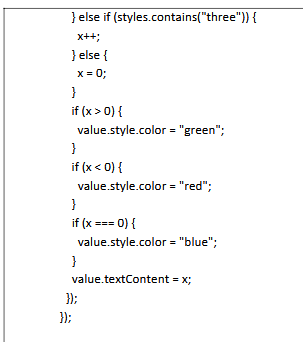
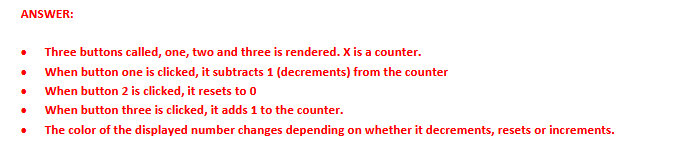
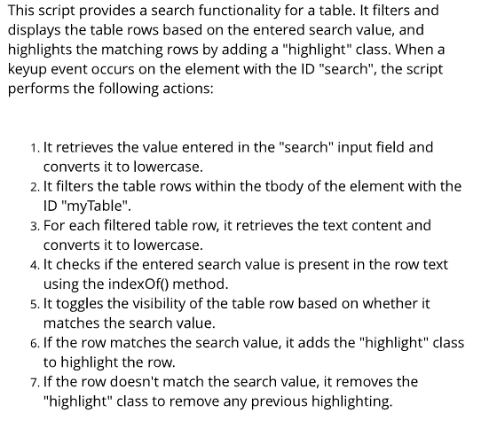
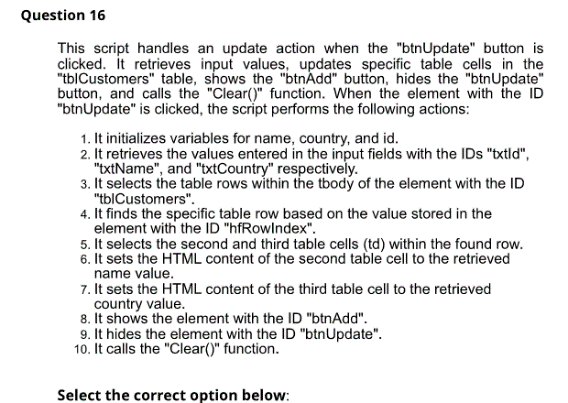
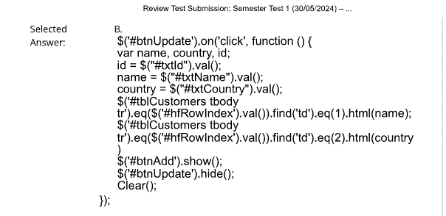
QUESTION 2 – Javascript file

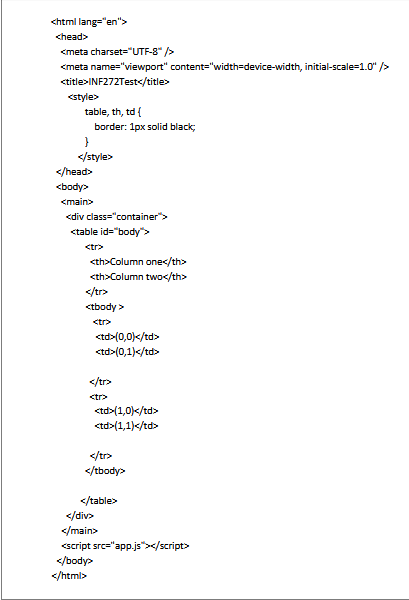
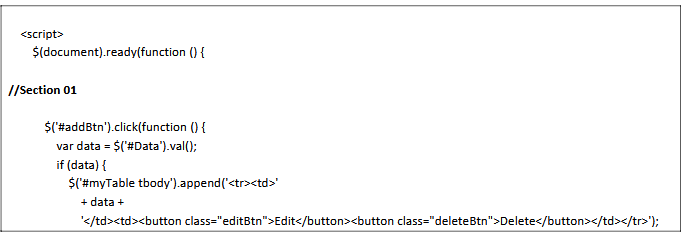
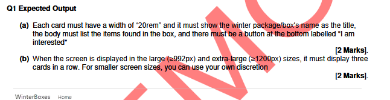
*When Play is clicked, a random circle brightens (either green, yellow, or red). This is done by changing the background colour to a lighter shade. We suggest the following: for the yellow bulb, change it to “yellow”, the red bulb to “tomato” and the green bulb to “light green”. See the example to the right where the yellow circle was hit When Reset is clicked, all colours change back to the darker shades and the comment below the buttons is removed.*

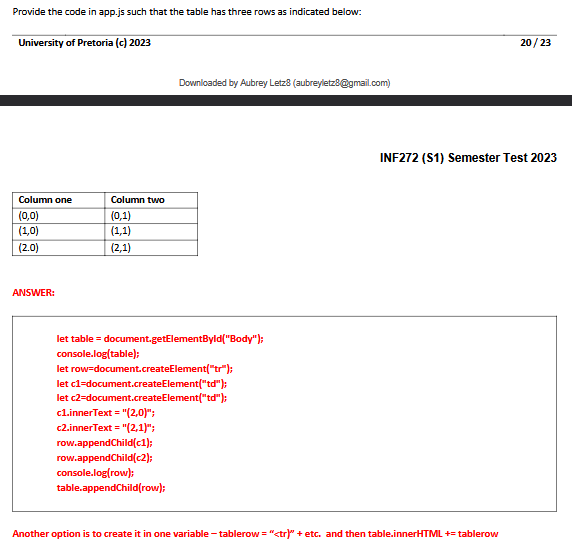
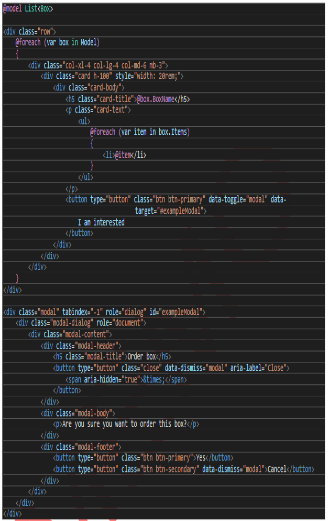
*When the red circle is hit, the word “Bullseye!” appears below the buttons as well as the number of times the user hit Bullseye at that stage. Please refer to the example provided*

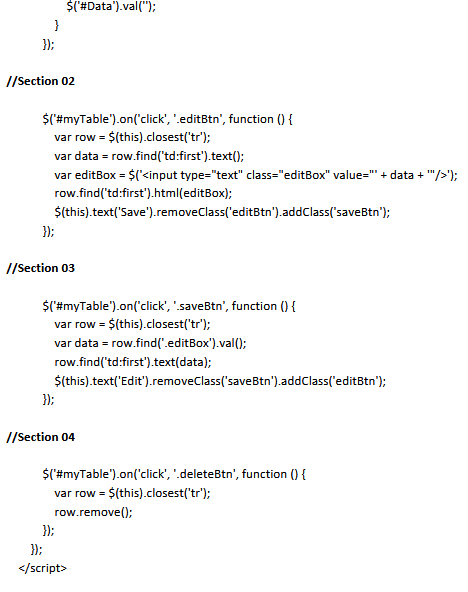
|  |  |  |
| --- | --- | --- |
| window.addEventListener("load", function () {  let startPlay = document.getElementById("play");  //add event listener to button  startPlay.addEventListener('click', function () {  play();  });  this.document.getElementById("reset").addEventListener('click',function(){  reset();  })  });  var count = 0;  function play() {  let randomNum = Math.floor(Math.random() \* 3);  typeof(points); | console.log(randomNum);  switch (randomNum) {  case 0:  {  document.getElementById("0").style.background = "lightgreen";  break;  }  case 1:  {  document.getElementById("1").style.background = "yellow";  break;  }  case 2:  { | document.getElementById("2").style.background = "tomato";  count += 1;  document.getElementById("message").innerHTML = "Bullseye!" + " " + count;  break;  }  default:  {  }  }  };  function reset() {  document.getElementById("0").style.background = "darkgreen";  document.getElementById("1").style.background = "goldenrod";  document.getElementById("2").style.background = "darkred";  document.getElementById("message").innerHTML = " ";  } |

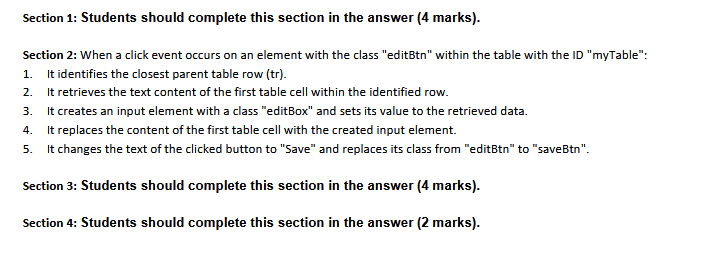


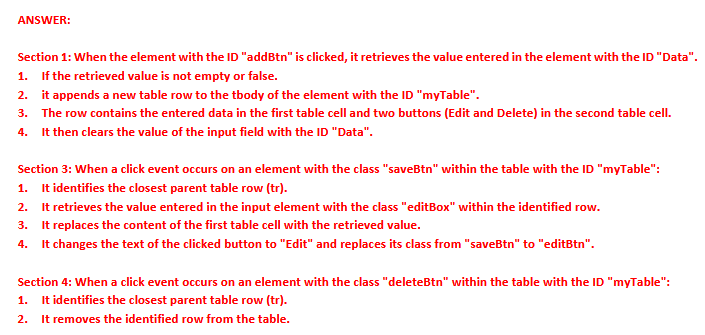












<

🡨------- ANSWERS

