Teleb, Jayson Emmanuel D. February 22, 2024

BSIT-32E2

Instructions: Read each question carefully and provide complete and clear answers. Avoid multiple-choice format responses. Focus on demonstrating your understanding through code, explanations, and discussions.

Part 1: C# (30 points)

(10 points) Write a C# program that calculates the area of a triangle given its base and height. Include user input for both values and display the calculated area.

```
1  using System;
2
3  class Program
4  {
5      static void Main(string[] args)
6      {
7          double baseLength, height, area;
8
9          Console.Write("Enter the base length of the triangle: ");
10         baseLength = Convert.ToDouble(Console.ReadLine());
11
12         Console.Write("Enter the height of the triangle: ");
13         height = Convert.ToDouble(Console.ReadLine());
14
15         area = 0.5 * baseLength * height;
16
17         Console.WriteLine($"The area of the triangle with base {baseLength} and height {height} is: {area}");
18
19         Console.ReadLine();
20     }
21 }
22
```

In this code, I declared a variable where it handles the length of the triangle, the height of triangle, and the area of triangle. In line 9 the user must input first the length of triangle, and in line 10, I use “Convert.ToDouble(Console.ReadLine())” where to get the values from user input. Then I declared another variable which is called “area” where it can handle the formula for the baseLength and height of the triangle. Lastly, in line 17, I use string interpolation to get the value of variables result.

(10 points) Declare an array of 5 integers and fill it with values based on a user-defined formula (e.g., n^2). Then, print the largest element in the array.

```
1  using System;
2
3  class Program
4  {
5      static void Main(string[] args)
6      {
7          // Declare an array of 5 integers
8          int[] array = {5, 4, 3, 2, 1};
9
10         // Fill the array with values based on a user-defined formula (n^2)
11         for (int i = 4; i < array.Length; i++)
12         {
13             int n = i + 1; // n starts from 1
14             array[i] = n * n;
15         }
16
17         // Find the largest element in the array
18         int max = array[4];
19         for (int i = 1; i < array.Length; i++)
20         {
21             if (array[i] > max)
22             {
23                 max = array[i];
24             }
25         }
26
27         // Print the largest element
28         Console.WriteLine("The largest element in the array is: " + max);
29     }
30 }
31
```

The first thing I to do is to declare an array integer which are 5, 4, 3, 2, 1 then fills the array with values based on a user-defined formula (n^2), starting from n = 5 (array index 4) up to the end of the array. So, it calculates n^2 and assigns it to each element from index 4 to the end of the array. Next, it finds the largest element in the array by iterating over each element and comparing it with the current maximum value (max). If a larger element is found, it updates max to that element. Finally, it prints out the largest element found.

(10 points) Implement a simple for loop that iterates from 1 to 10 and prints each number along with its square root.

```
using System;

class Program
{
    static void Main(string[] args)
    {
        for (int i = 1; i <= 10; i++)
        {
            int result = i*i;
            Console.WriteLine($"Square root of {i}: {result}");
        }
    }
}
```

In this code I declared for loop statement where it iteration (i) = 1 to 10, then I declared again a variables to determine the value of (i) then I use multiply it by itself, and I use string interpolation in output method to call the variable I use.

Part 2: HTML, CSS, and JavaScript (30 points)

HTML (10 points): You are provided with the following incomplete HTML code snippet:

```html
<!DOCTYPE html>
<html>
<head>
    <title>My Website</title>
    <style type="text/css">
        body{
            background-color: lightblue;
        }
        h1{
            color: red;
            padding: 20px;
        }
        h2{
            padding: 20px;
        }
        h3{
            padding: 20px;
        }
        p{
            font-size: 14px;
        }
        ol li{
            list-style-type: circle;
        }
    </style>
</head>
<body>
    <h1>This is the header 1</h1>
    <h2>This is the header 2</h2>
    <h3>This is the header 3</h3>
    <img src="">
    <p>This is paragraph</p>
    <p><a href=""></a></p>
    <ul>
        <li>Item 1</li>
        <li>Item 2</li>
    </ul>
    <ol>
        <li>Item 1</li>
        <li>Item 2</li>
        <li>Item 3</li>
    </ol>

</body>
</html>
```

Complete the code snippet by adding the following elements:

An image within the <body> tag with a relevant src attribute.

An ordered list (<ol>) with three items.

A hyperlink within a <p> tag that points to an external website.

A CSS styling rule using an inline style attribute to change the font color of the <h3> heading.

CSS (10 points): Create a CSS stylesheet that defines the following styles:

Change the background color of the body element to light blue.

Apply a padding of 20px to all headings (h1, h2, h3).

Set the font size of the <p> tag to 14px.

Make the list items (li) have a bullet point style instead of the default numbers.

JavaScript (10 points): Write a JavaScript function that takes a number as input and returns a string indicating whether the number is even or odd. Then, add a button to your HTML page that, when clicked, calls this function and displays the result (even or odd) in a paragraph element below the button.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title></title>
</head>
<body>
    <input type="number" id="numberInput" placeholder="Enter a number">
    <button onclick="checkEvenOrOdd()">Check Even or Odd</button>
    <p id="result"></p>

    <script>
        function checkEvenOrOdd() {
        var number = parseInt(document.getElementById("numberInput").value);

        if (isNaN(number)) {
            document.getElementById("result").innerText = "Invalid input. Please enter a valid number.";
            return;
        }

        if (number % 2 === 0) {
            document.getElementById("result").innerText = number + " is even.";
        } else {
            document.getElementById("result").innerText = number + " is odd.";
        }
    }
    </script>
</body>
</html>
```

When I enter a number, it will directly into an input field on the HTML page, and when I click the button, the JavaScript function will retrieve the value from the input field and perform the even/odd check accordingly.

Part 3: Essay Question (40 points)

Discuss the importance of object-oriented programming (OOP) concepts in software development. Explain the key principles of OOP (encapsulation, inheritance, polymorphism, abstraction) and provide examples of how they can be used to create more efficient, maintainable, and reusable code. Include real-world scenarios or cases where OOP is particularly valuable.

Object-Oriented Programming (OOP) is of paramount importance in software development as a whole. By embracing OOP principles such as modularity, encapsulation, inheritance, polymorphism, and abstraction, developers can create modular, maintainable, and scalable applications. OOP empowers developers to design and implement software systems more effectively, ensuring code reusability, extensibility, and collaboration. Understanding the significance of OOP to mastering the language and building robust and efficient applications.

The first key principle of OPP is **Encapsulation**, it involves grouping together data (such as attributes or properties) and methods (like functions or behaviors) that manipulate that data into a single unit, typically referred to as a class. It's akin to gathering data and its corresponding operations within a container, then managing access to that container. This approach shields an object's internal state from external interference, preserving the object's integrity and ensuring its state remains reliable. Ultimately, encapsulation enhances data security, maintains code integrity, and facilitates code upkeep.

Next, **Inheritance,** is a feature in object-oriented programming that enables a new class, known as a derived class or subclass, to inherit characteristics from an existing class, termed a base class or superclass. Through inheritance, the derived class can inherit attributes and methods from the base class, allowing for reuse of code. Additionally, the derived class can extend or modify these inherited properties and behaviors to introduce new functionalities or alter existing ones. This principle fosters code reuse, encourages modular design, and facilitates the establishment of hierarchical class relationships, where specialized classes inherit shared functionalities from broader, more general classes.

Then, the **Polymorphism**, deriving from the Greek words' "poly" meaning "many" and "morph" meaning "form," describes the capability of objects from different classes to be treated as instances of a shared superclass. This allows a single interface, like a method or function, to interact with objects of varying types, each exhibiting behavior specific to its type. Polymorphism typically occurs through method overriding, where subclasses provide their own implementation of a method, and method overloading, which involves defining multiple methods with the same name but different parameters. Embracing polymorphism enhances code's flexibility, extensibility, and adaptability, as it enables seamless interaction with diverse object types while maintaining a unified interface.

Lastly, **Abstraction**, its simplifying complex systems by concentrating on fundamental aspects while concealing extraneous details. In Object-Oriented Programming (OOP), abstraction is accomplished via abstract classes and interfaces. These constructs outline a framework for subclasses to adhere to without dictating implementation specifics. This approach enables the representation of real-world entities at a broader level, fostering improved code organization, modularity, and simplified maintenance. Abstraction aids in navigating complexity, fostering code reusability, and supporting design adaptability.

In real-world scenarios, OOP is particularly valuable in large-scale software development projects where the codebase needs to be modular, maintainable, and scalable. For example, in web development, OOP allows you to model web components as objects with well-defined interfaces, making it easier to manage complex interactions and updates across different parts of a website or web application. Similarly, in game development, OOP facilitates the creation of reusable game objects with encapsulated behaviors, enabling efficient development and maintenance of game systems. Overall, OOP promotes better code organization, reusability, and extensibility, leading to more efficient and maintainable software solutions.

Points Distribution:

Each part carries equal weight (30 points).

Code clarity, functionality, and explanations will be considered in grading.

The essay question focuses on understanding and application of OOP concepts.