**Villareal, Aubrey Mae A.**

**BSIT-32E1**

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.**

using System;

namespace TriangleAreaCalculator

{

class Calculator

{

static void Main(string[] args)

{

double baseLength, height, area;

//get baseLength

Console.Write("Enter the base of the triangle: ");

if (!double.TryParse(Console.ReadLine(), out baseLength) || baseLength <= 0)

{

Console.WriteLine("Please enter a valid POSITIVE number for the base length.");

return;

}

// get height

Console.Write("Enter the height of the triangle: ");

if (!double.TryParse(Console.ReadLine(), out height) || height <= 0)

{

Console.WriteLine("Please enter a valid POSITIVE number for the height.");

return;

}

// Calculate area

area = 0.5 \* baseLength \* height;

Console.WriteLine($"The Area of the triangle is: {area}");

Console.ReadKey();

}

}

}

**(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.**

using System;

namespace LargestElementInArray

{

class Compare

{

static void Main(string[] args)

{

int[] Array1 = { 15, 6, 10, 8, 35 };

Console.WriteLine("Numbers within the Array: ");

foreach (int num in Array1)

{

Console.WriteLine(num);

}

int bigger = Array1[0];

for (int i = 1; i < Array1.Length; i++)

{

if (Array1[i] > bigger)

{

bigger = Array1[i];

}

}

Console.WriteLine("The largest element in the array is: " + bigger);

Console.ReadKey();

}

}

}

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

using System;

namespace IterationLoop

{

class Loop

{

static void Main(string[] args)

{

int num = 1;

double sqr;

for (int i = 1; i <= 10; i++)

{

sqr = Math.Sqrt(num);

Console.WriteLine(num + " " + sqr);

num = num + 1;

}

}

}

}

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

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

<!DOCTYPE html>

<html>

<head>

<title>My Website</title>

<style>

body {

background-color: beige;

}

h1, h2, h3 {

padding: 20px;

}

h3{

color: brown;

}

p {

font-size: 14px;

}

li {

list-style-type: disc;

}

</style>

</head>

<body>

<h1>Welcome to...</h1>

<p>This is a paragraph...</p>

<img src="Avril.jpg">

<ol>

<li>WISH YOU WERE HERE</li>

<li>COMPLICATED</li>

<li>SKATER BOY</li>

</ol>

<p>Have a Soundtrip <a href="https://youtu.be/K1D5IpR4hk4?si=lpnJxlB6Gbp96j6e"> Watch my IDOL! </a></p>

<h3> Avril Lavigne is an Icon!</h3>

</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.

<button onclick="checkNumber()">Check Number</button>

<p id="result"></p>

<script>

function checkNumber() {

var number = prompt("Enter number:");

if (isNaN(number)) {

document.getElementById("result").textContent = "Invalid input. Please enter a valid number.";

return;

}

number = Number(number);

if (number % 2 === 0) {

document.getElementById("result").textContent = number + " is even.";

} else {

document.getElementById("result").textContent = number + " is odd.";

}

}

</script>

**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.

* OOP promotes collaboration among developers by providing a standardized approach to software development. The use of classes, objects, and well-defined interfaces facilitates teamwork, as different developers can work on different components independently while ensuring seamless integration. Benefits of OOPs is that it empowers programmers to treat objects from classes as instances of a shared superclass or interface. In Java, polymorphism enables the use of generic code, eliminating the need for explicit type checks.

**Encapsulation** is the process of combining methods (functions) that manipulate data with data (attributes) into a single unit or class. Through clearly defined interfaces, it only exposes the functionalities that are absolutely necessary, keeping the object's internal state hidden from the outside world. Encapsulation creates a distinct division between the implementation and interface, which facilitates data hiding and lowers system complexity.

A class (subclass or derived class) can inherit characteristics (methods) and behaviors (superclass or base class) from other classes through the technique of **Inheritance**. By extending or changing the superclass's functionality, the subclass encourages code reuse and creates a hierarchical relationship between classes. Because inheritance makes it easier to create specialized classes based on pre-existing ones, it helps keep code organized and maintained.

It is possible to regard objects of different classes as belonging to the same superclass thanks to **Polymorphism**. It does this by giving an interface the ability to perform the same operation in various ways, therefore allowing it to represent multiple underlying forms (classes). The two most common ways to create polymorphism are through overloading and overriding methods (redefining a method in a subclass) and defining numerous methods with the same name but different parameters. In object-oriented systems, polymorphism improves extensibility, modularity, and adaptability.

The process of determining an object's core qualities and disregarding its unimportant aspects is known as **Abstraction**. Its main goal is to create a simplified model that encapsulates the key characteristics of actual entities. With the use of abstraction, programmers may build models with more levels of detail comprehension, which helps them efficiently handle complexity and work with large-scale systems. Abstract classes and interfaces, which provide a framework for classes without giving full implementation details, are how OOP achieves abstraction.

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.