**Name:** Bayrante, Aliah Joy C.

**Course/Section:** BSIT 32E3

**Sample Assessment for Introduction to Programming**

This assessment is designed to evaluate your understanding of basic programming concepts in C#, HTML, CSS, and JavaScript.

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.

**Code:**

using System;

class Program

{

static void Main()

{

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

double baseValue = Convert.ToDouble(Console.ReadLine());

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

double heightValue = Convert.ToDouble(Console.ReadLine());

double area = (baseValue \* heightValue) / 2;

Console.WriteLine("The area of the triangle is: " + area);

}

}

**(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.**

**Code:**

using System;

class Program

{

static void Main()

{

int[] array = new int[5];

Console.WriteLine("Enter 5 numbers:");

for (int i = 0; i < 5; i++)

{

Console.Write("Number " + (i + 1) + ": ");

array[i] = Convert.ToInt32(Math.Pow(Convert.ToDouble(Console.ReadLine()), 2));

}

int max = array[0];

for (int i = 1; i < 5; i++)

{

if (array[i] > max)

{

max = array[i];

}

}

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

}

}

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

**Code:**

using System;

class Program

{

static void Main()

{

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

{

double sqrt = Math.Sqrt(i);

Console.WriteLine(i + " is a number from 1 to 10 and its square root is " + sqrt);

}

}

}

**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 First Web Page</title>

</head>

<body>

<h1>Welcome to my web page!</h1>

<p>This is a paragraph with some text.</p>

<ul>

<li>First item</li>

<li>Second item</li>

<li>Third item</li>

</ul>

<form action="https://www.example.com/submit\_form" method="post">

<label for="name">Name:</label><br>

<input type="text" id="name" name="name"><br>

<label for="email">Email:</label><br>

<input type="email" id="email" name="email"><br>

<input type="submit" value="Submit">

</form>

</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.**

**Code:**

<!DOCTYPE html>

<html>

<head>

<title>My Web Page</title>

</head>

<body>

<h1>Welcome to My Web Page</h1>

<p>Here is a picture:</p>

<img src="https://example.com/image.jpg" alt="An image">

<h2>My Favorite Things</h2>

<ol>

<li>Thing 1</li>

<li>Thing 2</li>

<li>Thing 3</li>

</ol>

<h3 style="color: red;">External Links</h3>

<p><a href="https://www.example.com/" target="\_blank">Example Website</a></p>

</body>

</html>

**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.**

**Code:**

body {

background-color: lightblue;

}

h1, h2, h3 {

padding: 20px;

}

p {

font-size: 14px;

}

li {

list-style-type: disc;

}

**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.

**Code:**

Javascript

function isEvenOrOdd(num) {

if (num % 2 === 0) {

return "even";

} else {

return "odd";

}

}

Html

<!DOCTYPE html>

<html>

<head>

<title>Even or Odd</title>

</head>

<body>

<button id="checkButton">Check if number is even or odd</button>

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

<script>

function isEvenOrOdd(num) {

if (num % 2 === 0) {

return "even";

} else {

return "odd";

}

}

document.getElementById("checkButton").addEventListener("click", function() {

var num = prompt("Enter a number");

var result = isEvenOrOdd(num);

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

});

</script>

</body>

</html>

**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.

answer

Object-oriented programming (OOP) is a programming paradigm that is based on the concept of "objects", which can contain data and methods that operate on that data. OOP is widely used in software development because it provides a way to organize and structure code in a way that is easy to understand, maintain, and extend. Here are some of the key principles of OOP and how they can be used to create more efficient, maintainable, and reusable code:

1. Encapsulation: Encapsulation is the practice of hiding the implementation details of an object from the outside world. This means that the data and methods of an object are kept private, and can only be accessed through a public interface. Encapsulation helps to prevent unauthorized access to the data and methods of an object, and makes it easier to change the implementation of an object without affecting other parts of the code. For example, a Person class might encapsulate the data and methods related to a person, such as their name, age, and a method to calculate their age in years. By encapsulating this information, the Person class can ensure that the name and age of a person cannot be modified directly, but only through the public methods provided by the class.
2. Inheritance: Inheritance is the practice of creating a new class that is based on an existing class. The new class inherits all the data and methods of the existing class, and can also add new data and methods of its own. Inheritance helps to promote code reuse, as common functionality can be implemented in a base class and then inherited by multiple derived classes. For example, a Shape class might define the data and methods related to a shape, such as its area and perimeter. A Circle class could then inherit from the Shape class, and add new data and methods related to circles, such as the radius and a method to calculate the circumference.
3. Polymorphism: Polymorphism is the practice of using a single interface to represent multiple types of objects. This means that an object can be treated as a generic type, and the specific implementation of that type is determined at runtime. Polymorphism helps to promote flexibility and extensibility in code, as new types of objects can be added to the system without affecting existing code. For example, a Shape class might define a method called calculateArea() that calculates the area of a shape. A Circle class and a Rectangle class could both inherit from the Shape class, and each class would provide its own implementation of the calculateArea() method. At runtime, the specific implementation of the calculateArea() method that is called would depend on the type of the object.
4. Abstraction: Abstraction is the practice of hiding the complexity of an object from the outside world. This means that an object can be represented by a simplified interface, which hides the underlying implementation details. Abstraction helps to promote modularity and maintainability in code, as changes to the implementation of an object do not affect other parts of the code. For example, a Car class might define a method called startEngine() that starts the engine of the car. The implementation of this method might involve complex interactions with the car's electrical and mechanical systems. However, by abstracting this complexity behind a simple interface, the user of the Car class does not need to know the details of how the engine is started.

In real-world scenarios, OOP is particularly valuable in large-scale software development projects, where code is often developed and maintained by multiple developers over an extended period of time. By using OOP principles, developers can create code that is easy to understand, maintain, and extend, and that can be reused across multiple projects. For example, a banking application might use OOP to create classes that represent different types of bank accounts, such as checking accounts and savings accounts. Each class would encapsulate the data and methods related to the specific type of account, and would inherit from a base class that defines common functionality. By using inheritance and polymorphism, the banking application could create a flexible and extensible system that can handle different types of accounts and transactions.