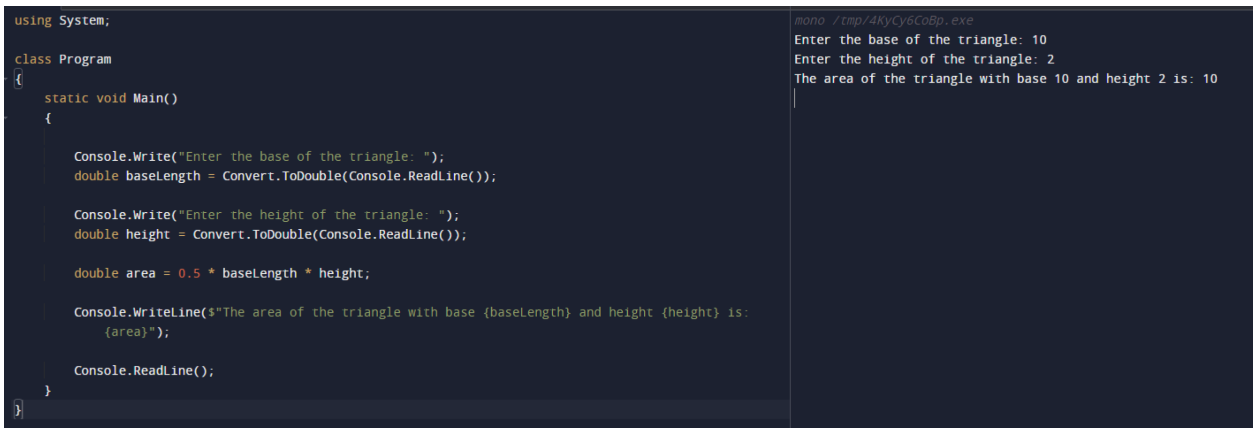
Part 1: C# (30 points)

(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.

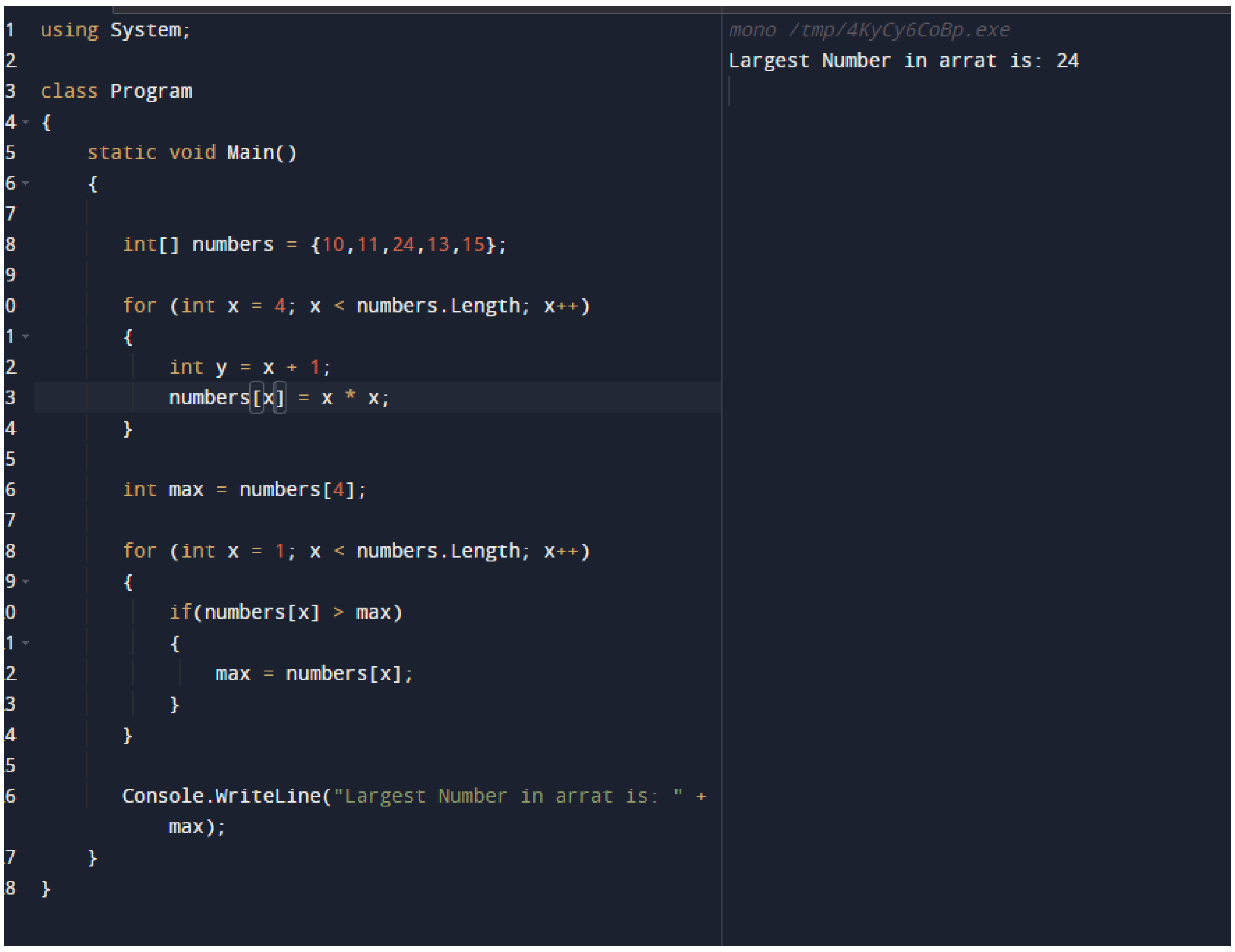


```csharp
using System;

class Program
{
    static void Main()
    {

        Console.Write("Enter the base of the triangle: ");
        double baseLength = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter the height of the triangle: ");
        double height = Convert.ToDouble(Console.ReadLine());

        double area = 0.5 * baseLength * height;

        Console.WriteLine($"The area of the triangle with base {baseLength} and height {height} is:
            {area}");

        Console.ReadLine();
    }
}
```

```
mono /tmp/4KyCy6CoBp.exe
Enter the base of the triangle: 10
Enter the height of the triangle: 2
The area of the triangle with base 10 and height 2 is: 10
```

- 1st the user will input the base and height of the triangle Reads the input values using Console.ReadLine() and converts them to doubles using Convert.ToDouble(). Next it the calculate the area of the triangle using this formula area = 0.5* baseLength * height, After computing the formula, Console.WriteLine will show the output/

(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.

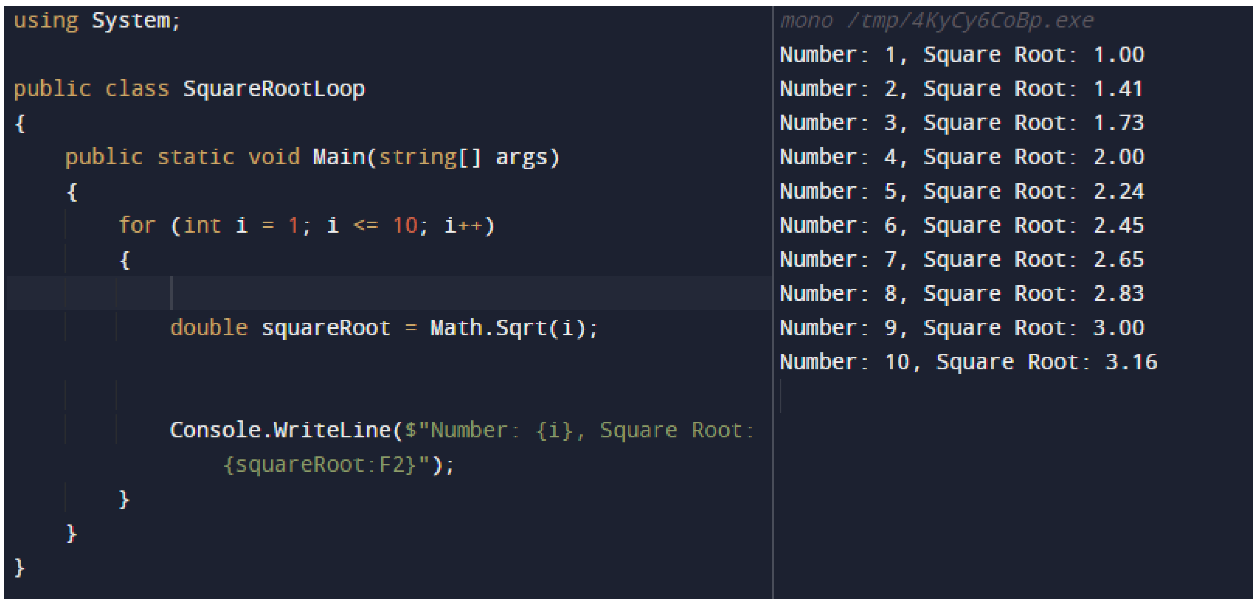


```csharp
using System;

class Program
{
    static void Main()
    {

        int[] numbers = {10,11,24,13,15};

        for (int x = 4; x < numbers.Length; x++)
        {
            int y = x + 1;
            numbers[x] = x * x;
        }

        int max = numbers[4];

        for (int x = 1; x < numbers.Length; x++)
        {
            if(numbers[x] > max)
            {
                max = numbers[x];
            }
        }

        Console.WriteLine("Largest Number in arrat is: " +
            max);
    }
}
```

```
mono /tmp/4KyCy6CoBp.exe
Largest Number in arrat is: 24
```

- The code defines an array named numbers = containing the values 10, 11, 24, 13, and 15. The For loop it iterates through each elements of the numbers array. Inside the Loop it compares the current element number[x] with variable max. After the loop finishes, the program prints the value of max, which now represents the largest number in the array.

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



```
using System;

public class SquareRootLoop
{
    public static void Main(string[] args)
    {
        for (int i = 1; i <= 10; i++)
        {

            double squareRoot = Math.Sqrt(i);


            Console.WriteLine($"Number: {i}, Square Root:
                {squareRoot:F2}");
        }
    }
}
```

```
mono /tmp/4KyCy6CoBp.exe
Number: 1, Square Root: 1.00
Number: 2, Square Root: 1.41
Number: 3, Square Root: 1.73
Number: 4, Square Root: 2.00
Number: 5, Square Root: 2.24
Number: 6, Square Root: 2.45
Number: 7, Square Root: 2.65
Number: 8, Square Root: 2.83
Number: 9, Square Root: 3.00
Number: 10, Square Root: 3.16
```

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>

</head>

<body>

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

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

<ul>

<li>Item 1</li>

<li>Item 2</li>

</ul>

</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.

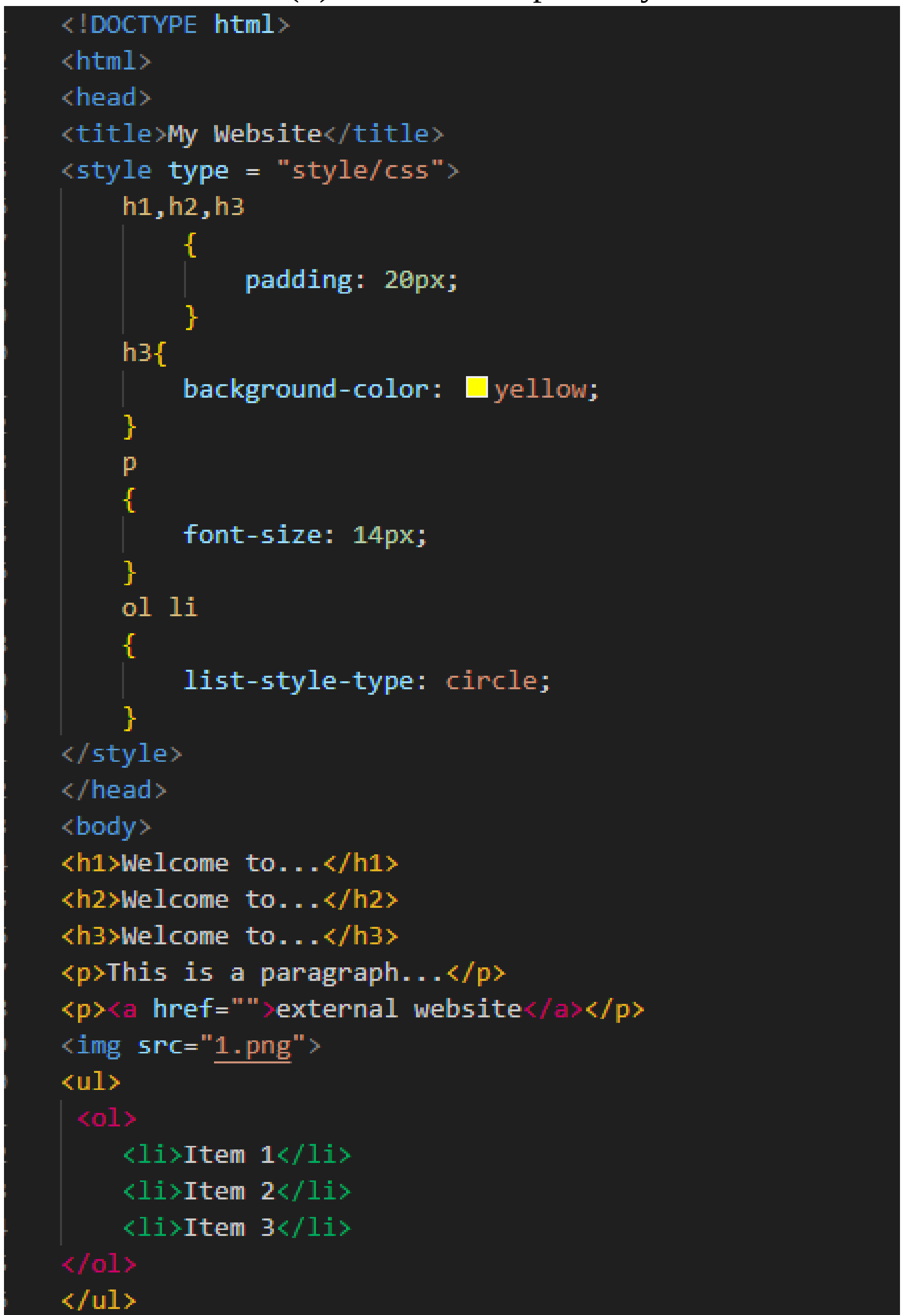
CSS (10 points): Create a CSS stylesheet that defines the following styles:

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

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

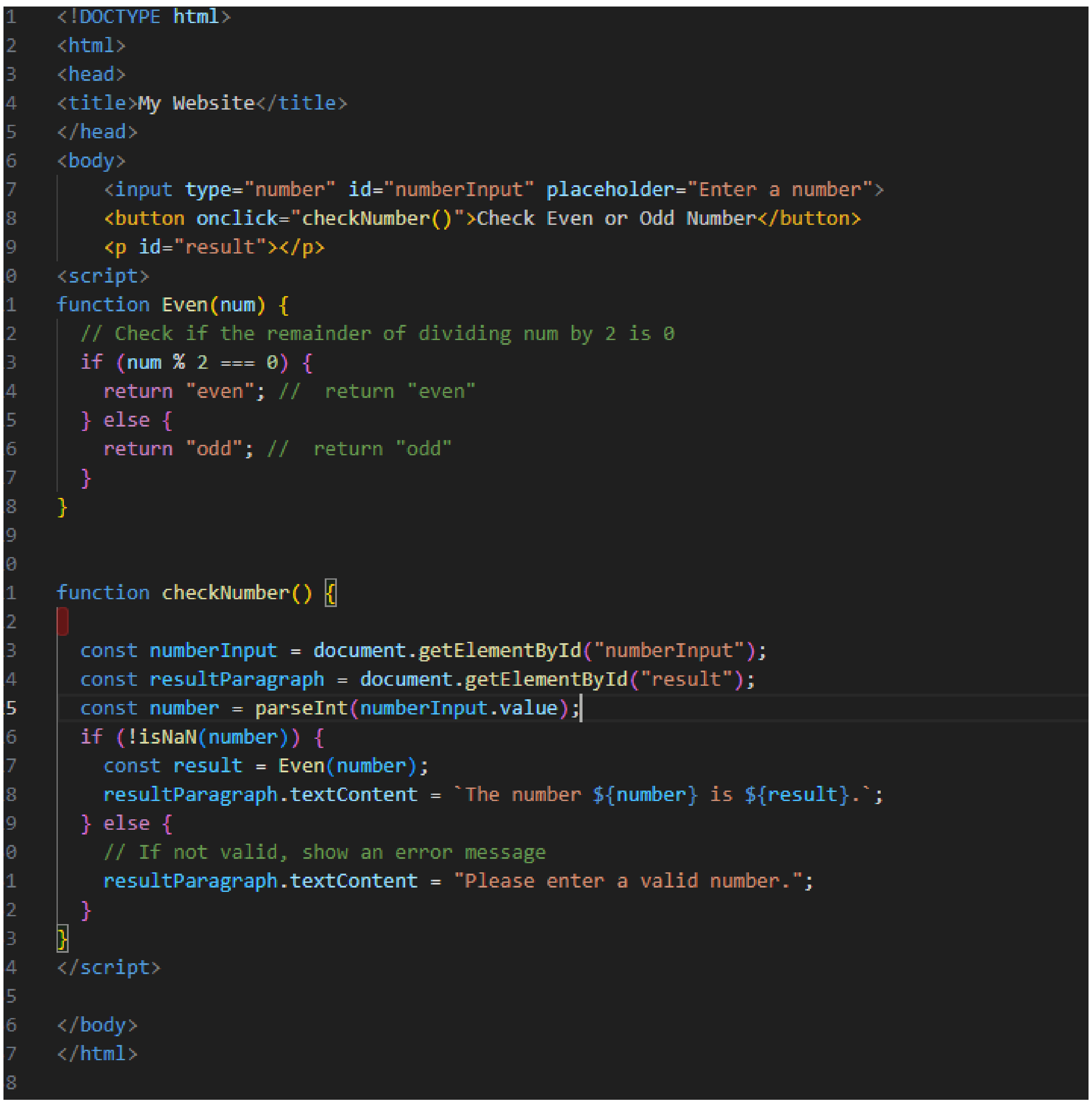
Set the font size of the <p> tag to 14px.

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



```
<!DOCTYPE html>
<html>
<head>
<title>My Website</title>
<style type = "style/css">
    h1,h2,h3
        {
            padding: 20px;
        }
    h3{
        background-color: yellow;
    }
    p
    {
        font-size: 14px;
    }
    ol li
    {
        list-style-type: circle;
    }
</style>
</head>
<body>
<h1>Welcome to...</h1>
<h2>Welcome to...</h2>
<h3>Welcome to...</h3>
<p>This is a paragraph...</p>
<p><a href="">external website</a></p>
<img src="1.png">
<ul>
 <ol>
    <li>Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
</ol>
</ul>
```

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.



```
<!DOCTYPE html>
<html>
<head>
<title>My Website</title>
</head>
<body>
    <input type="number" id="numberInput" placeholder="Enter a number">
    <button onclick="checkNumber()">Check Even or Odd Number</button>
    <p id="result"></p>
<script>
function Even(num) {
  // Check if the remainder of dividing num by 2 is 0
  if (num % 2 === 0) {
    return "even"; //  return "even"
  } else {
    return "odd"; //  return "odd"
  }
}


function checkNumber() {

  const numberInput = document.getElementById("numberInput");
  const resultParagraph = document.getElementById("result");
  const number = parseInt(numberInput.value);
  if (!isNaN(number)) {
    const result = Even(number);
    resultParagraph.textContent = `The number ${number} is ${result}.`;
  } else {
    // If not valid, show an error message
    resultParagraph.textContent = "Please enter a valid number.";
  }
}
</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.

Object-oriented programming (OOP) has become a fundamental paradigm in software development due to its ability to promote **efficiency, maintainability, and reusability**. It models real-world entities and their interactions, leading to code that's easier to understand, modify, and scale.

Principles of OOP

- **Encapsulation**

  is an idea that aims to keep the outside world from knowing how an object is implemented. It claims that all relevant data is inside to the object and that only a subset of the data is accessible outside. The internal workings and state of each object are kept confidentially within the designated class; other objects cannot access or modify it. Rather, they are limited to using a select few public services or features. This type of data concealing lowers the possibility of errors, increases program understandability, and gives program security and control over object state changes.

- **Inheritance**

  is a concept that lets programmers build new classes by extending or replacing the properties and methods of pre-existing (parent) classes. This helps to avoid code duplication and makes maintenance easier for projects with thousands of lines of code. Developers can construct objects with similar code or logic but distinct properties by using the parent class's logic in the child class. As a result, the code becomes less complex and there is no longer a need to construct a new object for every object that the program uses.

- **Polymorphism**

  is a concept that supports inheritance by enabling objects of different classes to use different codes to carry out identical activities. For example, a variety of information about objects of the "car," "plane," or "ship" type can be displayed using the "show information" method. Moreover, polymorphism facilitates the development of modular and flexible programs. In general, it makes development easier since it makes it possible to create shared functions and procedures that can be used to a variety of object kinds.

- **Abstraction**

  aids in concentrating on the key components of a system while ignoring the minor details that don't affect its main functions. It enables you to create programs that are easier to understand. You can think of abstraction as an extension of encapsulation. Consider applications that have thousands of lines of code. By virtue of the principle of abstraction, every item discloses only a particular mechanism of use. As a result, the internal code gains significant object independence. For example, in a movie database software, you may define a class called "Movie" that exposes only the most critical information—title, release year, and genre—while concealing the less critical features—shots or technical details.

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.