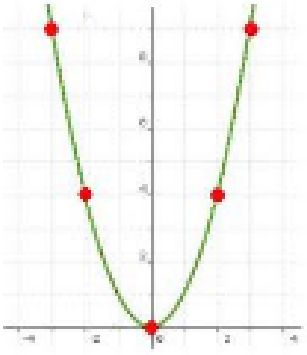
Ejercicio 2.4.1.

Queremos una aplicación web que permita resolver ecuaciones cuadráticas. Para ello indicaremos como parámetros los valores de los coeficientes a, b y c (números enteros).



$$ax^2 + bx + c = 0$$

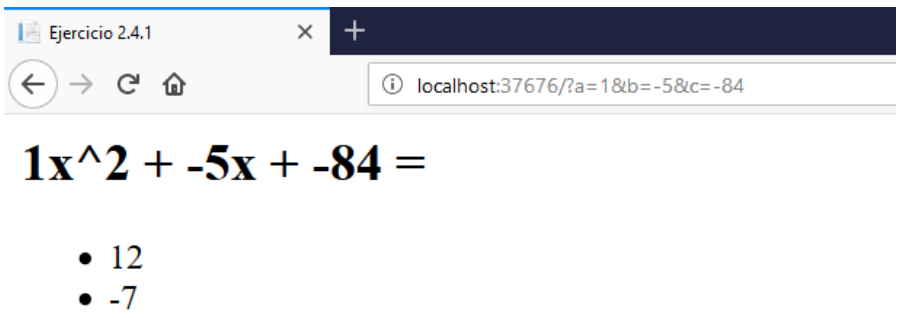
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Para el cálculo de la solución tendremos en cuenta los siguientes casos:

a. Si a==0, b==0, mostrará un mensaje diciendo que la ecuación es degenerada.
b. Si a==0 y b!=0, se mostrará el único resultado cuyo valor es -c/b.
c. En el resto de los casos, tenemos 2 soluciones que se mostrarán en la página de salida.

Pista: Una posible solución podría ser que el controlador devuelva los resultados a la vista en una colección. En la vista se podría comprobar el número de resultados que contiene la solución, de forma que ya sabríamos si tenemos dos, una o ninguna solución a la ecuación.

A continuación se muestra un ejemplo de ejecución:



Solución.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(int a, int b, int c)
        {
            ViewData["a"] = a;
            ViewData["b"] = b;
            ViewData["c"] = c;
            ViewData["result"] = Quadratic(a, b, c);

            return View();
        }

        private List<double> Quadratic(int a, int b, int c)
        {
            List<double> result = new List<double>();

            if(a == 0)
            {
                if(b != 0)
                {
                    result.Add(-c / (double)b);
                }
            }
            else
            {
                double t1 = Math.Sqrt(b * b - 4 * a * c);
                result.Add((-b + t1) / (2 * a));
                result.Add((-b - t1) / (2 * a));
            }

            return result;
        }
    }
}
```

Index.cshtml

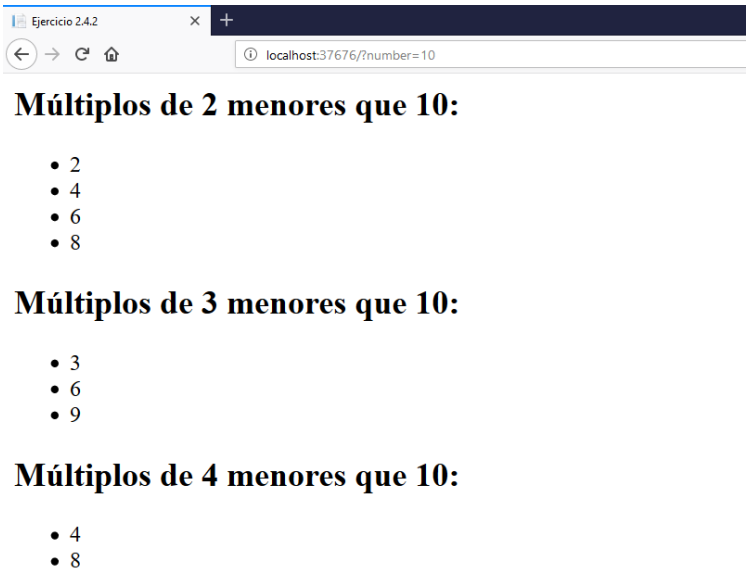
```
@using System.Collections.Generic

<html>
<head>
    <title>Ejercicio 2.4.1</title>
</head>
<body>
    <h2>@ViewData["a"]x^2 + @ViewData["b"]x + @ViewData["c"] = </h2>
    <ul>
        @{
            var result = ViewData["result"] as List<double>;
            if (result.Count == 0)
            {
                <li>Ecuación degenerada, no hay solución</li>
            }
            else
            {
                foreach (var n in result)
                {
                    <li>@n</li>
                }
            }
        }
    </ul>
</body>
</html>
```

Ejercicio 2.4.2.

En este caso queremos una aplicación que a partir de un número indicado por el usuario:

1) Invoque a una función para que calcule todos os números pares (múltiplos de 2) menores que el número indicado.
2) Invoque a la misma función, para que esta vez calcule todos los múltiplos de 3 menores que el número indicado.
3) Invoque de nuevo la misma función, para calcular los múltiplos de 4 menores que el número indicado.

Por último, todos estos números serán mostrados en una página en el navegador.



Solución.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;
using System.Collections.Generic;

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(int number)
        {
            ViewData["limit"] = number;
            ViewData["multsOf2"] = LowerMults(number, 2).ToArray();
            ViewData["multsOf3"] = LowerMults(number, 3).ToArray();
            ViewData["multsOf4"] = LowerMults(number, 4).ToArray();

            return View();
        }

        private List<int> LowerMults(int limit, int m)
        {
            List<int> result = new List<int>();

            for (int n = 1; n < limit; n++)
            {
                if (n % m == 0)
                {
                    result.Add(n);
                }
            }

            return result;
        }
    }
}
```

Index.cshtml

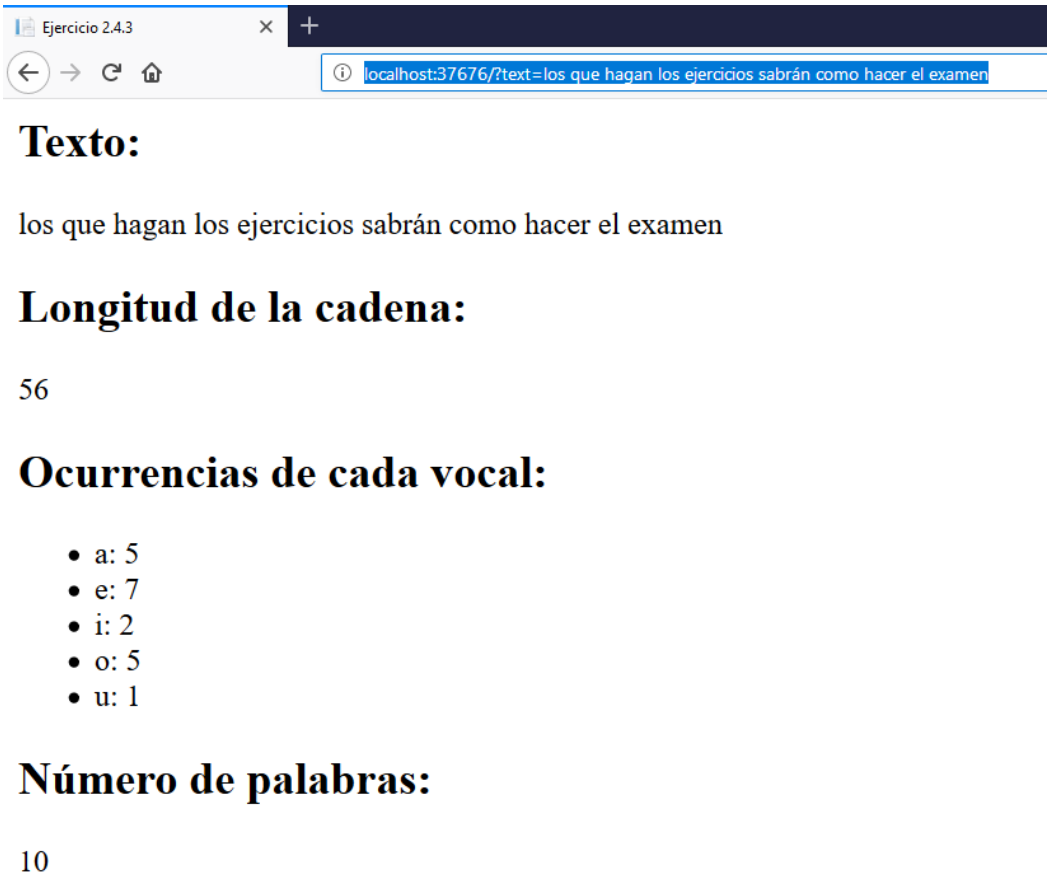
```
<html>
<head>
    <title>Ejercicio 2.4.2</title>
</head>
<body>
    <h2>Múltiplos de 2 menores que @ViewData["limit"]:</h2>
    <ul>
        @{
            foreach (var n in @ViewData["multsOf2"] as int[])
            {
                <li>@n</li>
            }
        }
    </ul>

    <h2>Múltiplos de 3 menores que @ViewData["limit"]:</h2>
    <ul>
        @{
            foreach (var n in @ViewData["multsOf3"] as int[])
            {
                <li>@n</li>
            }
        }
    </ul>

    <h2>Múltiplos de 4 menores que @ViewData["limit"]:</h2>
    <ul>
        @{
            foreach (var n in @ViewData["multsOf4"] as int[])
            {
                <li>@n</li>
            }
        }
    </ul>
</body>
</html>
```

Ejercicio 2.4.3.

En este ejercicio realizaremos una serie de tareas con una cadena de caracteres. Para ello el usuario introducirá una cadena de texto como parámetro en la url.

A continuación la página de salida debe mostrar una serie de datos sobre la cadena. Se enumera la información que se debe mostrar:

- La propia cadena de texto.
- Número de caracteres de la cadena o longitud.
- Número de ocurrencias de cada vocal (sin distinguir mayúsculas de minúsculas) en la cadena.
- Número de palabras de la cadena (se considera que las palabras están separadas por un espacio en blanco).



Solución.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(string text)
        {
            ViewData["text"] = text;

            ViewData["length"] = text.Length;

            int noa = 0;
            int noe = 0;
            int noi = 0;
            int noo = 0;
            int nou = 0;
            foreach (var c in text.ToLower())
            {
                switch (c)
                {
                    case 'a':
                        noa++;
                        break;
                    case 'e':
                        noe++;
                        break;
                    case 'i':
                        noi++;
                        break;
                    case 'o':
                        noo++;
                        break;
                    case 'u':
                        nou++;
                        break;
                }
            }
            ViewData["noa"] = noa;
            ViewData["noe"] = noe;
            ViewData["noi"] = noi;
            ViewData["noo"] = noo;
            ViewData["nou"] = nou;

            ViewData["words"] = text.Split(" ").Length;

            return View();
        }
    }
}
```

Index.cshtml

```
<html>
<head>
    <title>Ejercicio 2.4.3</title>
</head>
<body>
    <h2>Texto:</h2>
    @ViewData["text"]

    <h2>Longitud de la cadena:</h2>
    @ViewData["length"]

    <h2>Ocurrencias de cada vocal:</h2>
    <ul>
        <li>a: @ViewData["noa"]</li>
        <li>e: @ViewData["noe"]</li>
        <li>i: @ViewData["noi"]</li>
        <li>o: @ViewData["noo"]</li>
        <li>u: @ViewData["nou"]</li>
    </ul>

    <h2>Número de palabras:</h2>
    @ViewData["words"]
</body>
</html>
```

Ejercicio 2.4.4.

Para la realización de este último ejercicio esperaremos también una cadena como parámetro. En nuestra página se mostrará el resultado que se obtiene cada vez que se realiza una rotación de un carácter a la derecha sobre dicha cadena, hasta que se obtenga de nuevo la cadena original.

Solución.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace Ejercicio.Controllers
{
    public class HomeController : Controller
    {
        private static int count = 0;

        public IActionResult Index(string text)
        {
            string[] rotations = new string[text.Length];

            for (int r = 0; r < text.Length; r++)
            {
                rotations[r] = text.Substring(text.Length - r - 1)
                    + text.Substring(0, text.Length - r - 1);
            }

            ViewData["text"] = text;
            ViewData["rotations"] = rotations;

            return View();
        }
    }
}
```

Index.cshtml

```
<html>
<head>
    <title>Ejercicio 2.4.4</title>
</head>
<body>
    <h2>Rotaciones del texto:</h2>
    <ul>
        <li>@ViewData["text"]</li>
        @foreach (var rotation in ViewData["rotations"] as string[])
        {
            <li>@rotation</li>
        }
    </ul>
</body>
</html>
```