1. Differentiate between .NET Framework and .NET Core.

   Answer:

   Differences between .NET Framework and .NET Core are listed below:

.NET Framework	**.NET Core**
Only certain components of .NET Framework are open source.	.NET Core is an open source
On the Windows operating system, the.NET Framework is installed as a single package.	Due to its cross-platform nature, .NET Core is packaged and deployed without regard to the underlying operating system.
While REST API services are supported by .Net Framework, microservice creation and implementation are not.	Micro-services can be created and implemented using .Net Core, and a REST API must be created by the user in order to do so.
.NET Framework is not very effective in terms of performance and scalability.	Compared to .NET Framework, it offers high performance and scalability.
Its compatibility is only limited to Windows operating system.	It is compatible with different operating system like Windows, Linux, and Mac OS.

2. Write about .NET6 and list its features.

   Answer:

   .NET 6 is a unified development platform that enables programmers to create cloud, browser, desktop, mobile, gaming, IoT and AI applications. All of these sub platforms share the same basic class libraries, APIs, and underlying infrastructure, including languages and compilers. Features of .NET6 are:

   - It has improved performance, which makes it faster and more efficient.
   - It has the ability to build native applications for Windows, Mac OS and Linux.

- It includes many improvements to ASP.NET Core building features like HTTP/3 support, gRPC Web, api.
- It helps to create cross-platform desktop application using technologies like WinUI and MAUI.
- With .NET6, we can create single-file applications that contain all the necessary components and libraires, making it easier to deploy and distribute to applications.
- It offers improved support for containerized applications, including faster startup times and better integration with container orchestration platforms like Kubernetes.

3. Build a simple form that allows users to enter their personal information using ASP.NET web forms.

Requirements:

The form should have a user interface that includes fields for the user's name, address, email, phone number, and date of birth.

The form should use client-side validation to ensure that the user's input is valid.

The form should have a "Submit" button that displays a confirmation message when clicked.

Hints:

- Use ASP.NET web form controls, such as TextBox and Button, to create the user interface.
- Use the RegularExpressionValidator control to validate the user's input for fields such as email and phone number.
- Use the RequiredFieldValidator control to ensure that all required fields are filled in.
- Use the ValidationSummary control to display error messages on the page.
- Use the OnClick event of the "Submit" button to display a confirmation message.

Answer:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="MyForm.aspx.cs"
Inherits="MyApplication.MyForm" %>


<!DOCTYPE html>
```

```
<html>
<head>
  <title>My Form</title>
</head>
<body>
  <h1>Personal Information</h1>
  <form id="myForm" runat="server">
    <div>
      <label for="name">Name:</label>
      <asp:TextBox ID="name" runat="server"></asp:TextBox>
      <asp:RequiredFieldValidator ID="nameValidator" ControlToValidate="name"
ErrorMessage="Name is required" runat="server" />
    </div>
    <div>
      <label for="address">Address:</label>
      <asp:TextBox ID="address" runat="server"></asp:TextBox>
    </div>
    <div>
      <label for="email">Email:</label>
      <asp:TextBox ID="email" runat="server"></asp:TextBox>
      <asp:RegularExpressionValidator ID="emailValidator" ControlToValidate="email"
ValidationExpression="^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$" ErrorMessage="Invalid email"
runat="server" />
    </div>
    <div>
      <label for="phone">Phone Number:</label>
      <asp:TextBox ID="phone" runat="server"></asp:TextBox>
      <asp:RegularExpressionValidator ID="phoneValidator" ControlToValidate="phone"
ValidationExpression="^\d{10}$" ErrorMessage="Invalid phone number" runat="server" />
    </div>
```

```
        <div>
            <label for="dob">Date of Birth:</label>
            <asp:TextBox ID="dob" runat="server"></asp:TextBox>
        </div>
        <asp:ValidationSummary ID="validationSummary" runat="server" />
        <asp:Button ID="submitButton" Text="Submit" OnClick="submitButton_Click" runat="server" />
    </form>
</body>
</html>
```

Form.aspx.cs:-

```
protected void submitButton_Click(object sender, EventArgs e)
{
    // Code to save the user's information to a database or send it via email
    Response.Write("Thank you for submitting your information!");
}
```