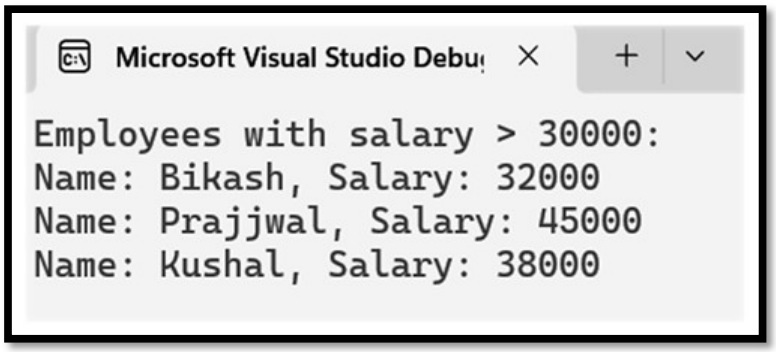
Lab: 1

Write a C# program to Create a class called Employee with properties: Id, Name, and Salary. Then add at least 5 employees to a list, Use **LINQ** to find all employees whose **salary is greater than 30000**,Display only the **Name** and **Salary** of those employees.

Code:

```
using System;
using System.Collections.Generic;
using System.Linq;
public class Employee
{
   public int Id { get; set; }
   public string Name { get; set; }
   public double Salary { get; set; }

   public Employee(int id, string name, double salary)
   {
      Id = id;
      Name = name;
      Salary = salary;
   }
}
class Program
{
   static void Main()
   {
      List<Employee> employees = new List<Employee>()
      {
         new Employee(1, "Dinesh", 25000),
         new Employee(2, "Bikash", 32000),
         new Employee(3, "Prajjwal", 45000),
         new Employee(4, "Sandesh", 28000),
         new Employee(5, "Kushal", 38000)
      };
 var highSalaryEmployees = employees
         .Where(emp => emp.Salary > 30000)
         .Select(emp => new { emp.Name, emp.Salary });
 Console.WriteLine("Employees with salary > 30000:");
      foreach (var emp in highSalaryEmployees)
      {
         Console.WriteLine($"Name: {emp.Name}, Salary: {emp.Salary}");
      }
   }
}
```

Output



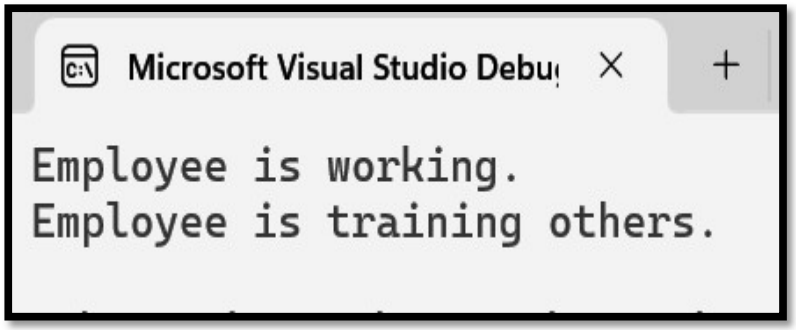
Lab: 2

Write a C# program to demonstrate multiple inheritance using interfaces.

Code:

```
using System;
interface IWorker
{
   void Work();
}
interface ITrainer
{
   void Train();
}
class Employee : IWorker, ITrainer
{
   public void Work()
   {
      Console.WriteLine("Employee is working.");
   }
   public void Train()
   {
      Console.WriteLine("Employee is training others.");
   }
}
class Program
{
   static void Main()
   {
      Employee emp = new Employee();
      emp.Work();
      emp.Train();
   }
}
```

Output:



Lab: 3

Write an ASP.NET Core MVC application to create a model, pass its data from the controller, and display it in a Razor view.

Code:

1. Create a Model (Employee.cs)

```
public class Employee
{
   public int Id { get; set; }
   public string Name { get; set; }
   public double Salary { get; set; }
}
```

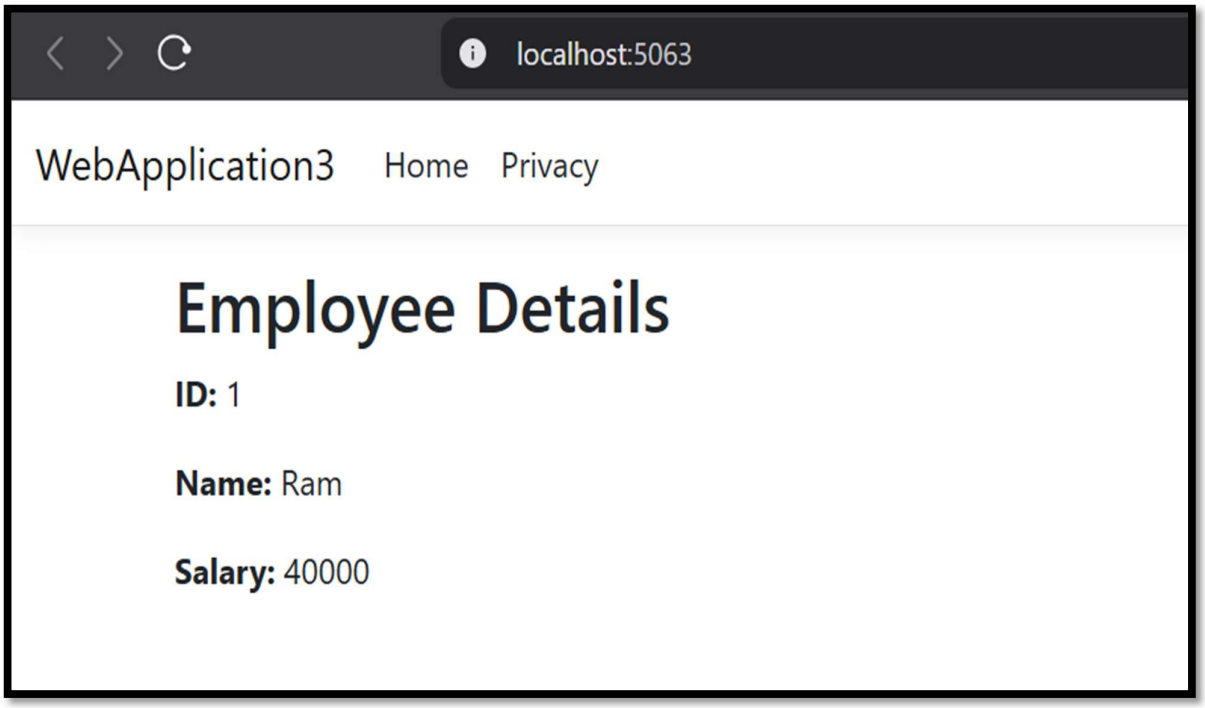
2. **Create a Controller (EmployeeController.cs)**

```
using Microsoft.AspNetCore.Mvc;
using YourProjectName.Models;
public class EmployeeController : Controller
{
   public IActionResult Index()
   {
      var emp = new Employee
      {
         Id = 1,
         Name = "Ram",
         Salary = 40000
      };
 return View(emp);
   }
}
```

3. **Create a Razor View (Views/Employee/Index.cshtml)**

```
@model YourProjectName.Models.Employee

<h2>Employee Details</h2>

<p><strong>ID:</strong> @Model.Id</p>

<p><strong>Name:</strong> @Model.Name</p>

<p><strong>Salary:</strong> @Model.Salary</p>
```

Output:


localhost:5063
WebApplication3
Home
Privacy
Employee Details
ID: 1
Name: Ram
Salary: 40000

Lab: 4

Write a program using jQuery to validate an HTML form with fields like name, email, and password.

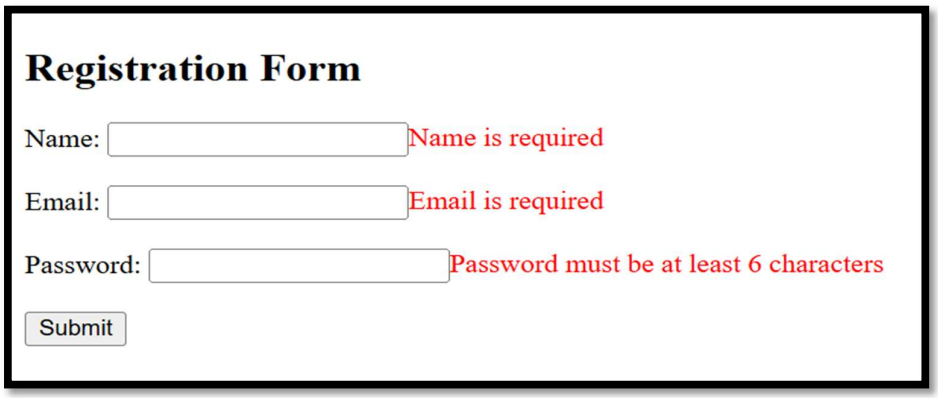
Code:

```
<!DOCTYPE html>
<html>
<head>
   <title>Form Validation with jQuery</title>
   <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
   <style>
      .error { color: red; }
   </style>
</head>
<body>

<h2>Registration Form</h2>

<form id="myForm">
   Name: <input type="text" id="name"><span class="error" id="nameError"></span><br><br>
   Email: <input type="text" id="email"><span class="error" id="emailError"></span><br><br>
   Password: <input type="password" id="password"><span class="error"
id="passwordError"></span><br><br>
   <button type="submit">Submit</button>
</form>

<script>
   $(document).ready(function () {
      $('#myForm').submit(function (e) {
         e.preventDefault();

         var isValid = true;
         $('.error').text(''); // Clear previous errors

         var name = $('#name').val().trim();
         var email = $('#email').val().trim();
         var password = $('#password').val().trim();

         if (name === '') {
            $('#nameError').text('Name is required');
            isValid = false;
         }

         if (email === '') {
            $('#emailError').text('Email is required');
```

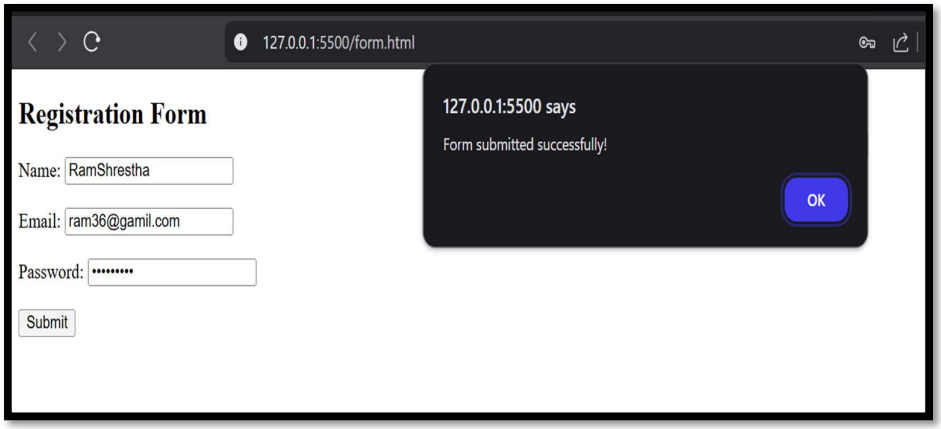
```
            isValid = false;
        } else if (!/\S+@\S+\.\S+/.test(email)) {
            $('#emailError').text('Invalid email format');
            isValid = false;
        }

        if (password.length < 6) {
            $('#passwordError').text('Password must be at least 6 characters');
            isValid = false;
        }

        if (isValid) {
            alert("Form submitted successfully!");
            // You can proceed to submit the form to server here
        }
    });
  });
</script>

</body>
</html>
```

Output:

When form is submitted empty:



After Entering valid input:



Lab: 5

Write a program in ASP.NET Core MVC to demonstrate the use of Session and Cookie by storing and displaying the user's name.

Steps:

1.Create a new ASP.NET Core MVC project.

2.Configure Session in Startup.cs or Program.cs.

3.Create a Controller to set and get Session and Cookie values.

4.Create Views to interact with the user.

1. Program.cs (for .NET 6/7/8):

csharp
CopyEdit

```
var builder = WebApplication.CreateBuilder(args);

// Add services
builder.Services.AddControllersWithViews();
builder.Services.AddSession(); // Add session service

var app = builder.Build();

app.UseStaticFiles();
app.UseRouting();
app.UseSession(); // Use session middleware

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

---

2. HomeController.cs:

csharp
CopyEdit

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Http;

namespace SessionCookieDemo.Controllers
{
    public class HomeController : Controller
    {
        public IActionResult Index()
        {
```

```
        return View();
    }

    [HttpPost]
    public IActionResult SetData(string username)
    {
        // Set session
        HttpContext.Session.SetString("Username", username);

        // Set cookie
        CookieOptions option = new CookieOptions();
        option.Expires = DateTime.Now.AddMinutes(30);
        Response.Cookies.Append("UsernameCookie", username, option);

        return RedirectToAction("ShowData");
    }

    public IActionResult ShowData()
    {
        ViewBag.SessionUser = HttpContext.Session.GetString("Username");
        ViewBag.CookieUser = Request.Cookies["UsernameCookie"];
        return View();
    }
  }
}
```

---

3. Index.cshtml (in Views/Home):

html
CopyEdit

```
@{
  ViewData["Title"] = "Index";
}

<h2>Enter Your Name</h2>

<form asp-action="SetData" method="post">
  <label>Name:</label>
  <input type="text" name="username" required />
  <button type="submit">Submit</button>
</form>
```

---

4. ShowData.cshtml (in Views/Home):

html
CopyEdit

```
@{
```

```
    ViewData["Title"] = "ShowData";
}

<h2>Session and Cookie Data</h2>

<p><strong>Session Username:</strong> @ViewBag.SessionUser</p>
<p><strong>Cookie Username:</strong> @ViewBag.CookieUser</p>
```

Output: