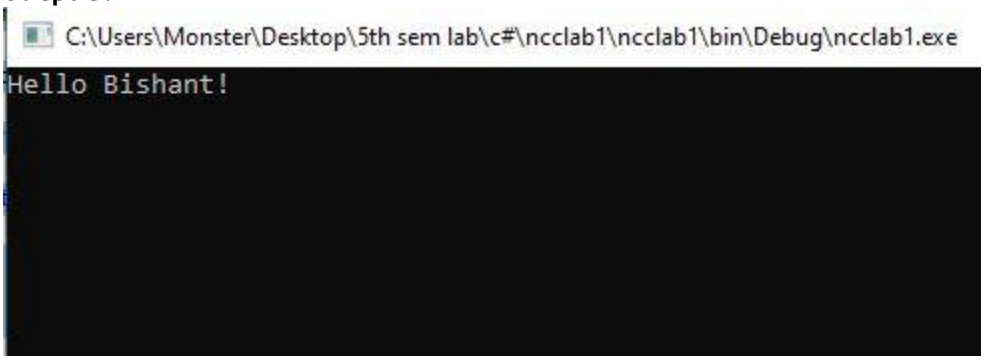
// Run a basic C# application with hello your name

```
using System;

namespace ncclab1
{
    class Program
    {
        static void Main(string[] args)
        {

            Console.WriteLine("Hello Bishant!");
            Console.ReadKey();
        }
    }
}
```

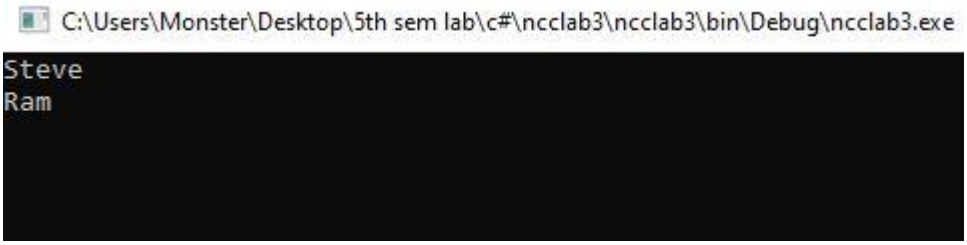
Output:



//c# program showing LINQ operation

```
using System;
using System.Collections.Generic;
using System.Linq;
namespace ncclab3
{
    public class Program
    {
        public static void Main()
        {
            IList<Student> studentList = new List<Student>() {
            new Student() { StudentID = 1, StudentName = "John",
                        Age = 18, StandardID = 1 } ,
                new Student() { StudentID = 2, StudentName = "Steve",
                        Age = 21, StandardID = 1 } ,
                new Student() { StudentID = 3, StudentName = "Bill",
                        Age = 18, StandardID = 2 } ,
                new Student() { StudentID = 4, StudentName = "Ram" ,
                        Age = 20, StandardID = 2 } ,
                new Student() { StudentID = 5, StudentName = "Ron" ,
                        Age = 21 }
        };
            var studentNames = studentList.Where(s => s.Age > 18)
                                .Select(s => s)
                                .Where(st => st.StandardID > 0)
                                .Select(s => s.StudentName);
            foreach (var name in studentNames)
            {
                Console.WriteLine(name);
                Console.ReadKey();
            }
        }
    }
    public class Student
    {
        public int StudentID { get; set; }
        public string StudentName { get; set; }
        public int Age { get; set; }
        public int StandardID { get; set; }
    }}
```

Output:



```
//c# program to show async I/O bound
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net.Http;
using System.Threading.Tasks;

namespace ncclab4
{
    class Program
    {
            private const string URL = "https://docs.microsoft.com/en-
      us/dotnet/csharp/csharp";
        static void Main(string[] args)
        {
            DoSynchronousWork();
            var someTask = DoSomethingAsync();
            DoSynchronousWorkAfterAwait();
            someTask.Wait(); //this is a blocking call, use it only on Main
            //method
            Console.ReadLine();
        }
        public static void DoSynchronousWork()
        {
            // You can do whatever work is needed here
            Console.WriteLine("1. Doing some work synchronously");
        }

            static async Task DoSomethingAsync() //A Task return type will
            //eventually yield a void
        {
            Console.WriteLine("2. Async task has started...");
            await GetStringAsync(); // we are awaiting the Async Method
            //GetStringAsync
        }

        static async Task GetStringAsync()
        {
            using (var httpClient = new HttpClient())
            {
                Console.WriteLine("3. Awaiting the result of GetStringAsync
                of Http Client...");
                string result = await httpClient.GetStringAsync(URL);
      //execution pauses here while awaiting GetStringAsync to complete

                //From this line and below, the execution will resume once
                //the above awaitable is done
                //using await keyword, it will do the magic of unwrapping the
                //Task<string> into string (result variable)
```

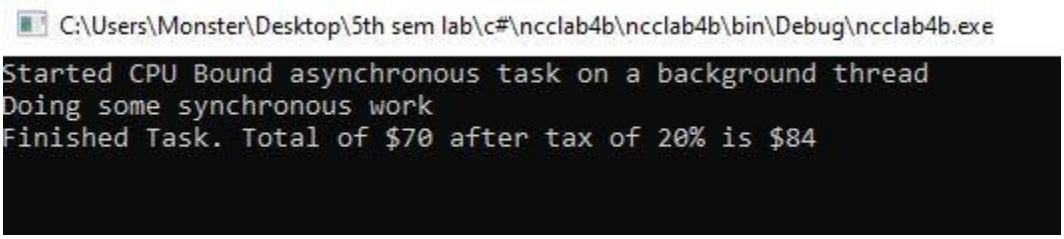
```
                Console.WriteLine("4. The awaited task has completed. Let's
                get the content length...");
                Console.WriteLine($"5. The length of http Get for {URL}");
                Console.WriteLine($"6. {result.Length} character");
            }
        }
        static void DoSynchronousWorkAfterAwait()
        {
             //This is the work we can do while waiting for the awaited Async
          //Task to complete
          Console.WriteLine("7. While waiting for the async task to finish,
          we can do some unrelated work...");
            for (var i = 0; i <= 5; i++)
            {
                for (var j = i; j <= 5; j++)
                {
                    Console.Write("*");
                }
                Console.WriteLine();
            }

        }
    }

}
```

Output:

```
C:\Users\Monster\Desktop\5th sem lab\c#\ncclab4\ncclab4\bin\Debug\ncclab4.exe
1. Doing some work synchronously
2. Async task has started...
3. Awaiting the result of GetStringAsync of Http Client...
7. While waiting for the async task to finish, we can do some unrelated work...
******
*****
****
***
**
*
4. The awaited task has completed. Let's get the content length...
5. The length of http Get for https://docs.microsoft.com/en-us/dotnet/csharp/csharp
6. 41015 character
```

```
//c# program async cpu bound
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ncclab4b
{
    class Program
    {
        static void Main(string[] args)
        {
            var totalAfterTax = CalculateTotalAfterTaxAsync(70);
            DoSomethingSynchronous();

            totalAfterTax.Wait();
            Console.ReadLine();
        }
        private static void DoSomethingSynchronous()
        {
            Console.WriteLine("Doing some synchronous work");
        }

        static async Task<float> CalculateTotalAfterTaxAsync(float value)
        {
             Console.WriteLine("Started CPU Bound asynchronous task on a
            background thread");
            var result = await Task.Run(() => value * 1.2f);
            Console.WriteLine($"Finished Task. Total of ${value} after tax of
      20% is ${result} ");
            return result;
        }
    }

}
```

Output:



// Basic asp.net application

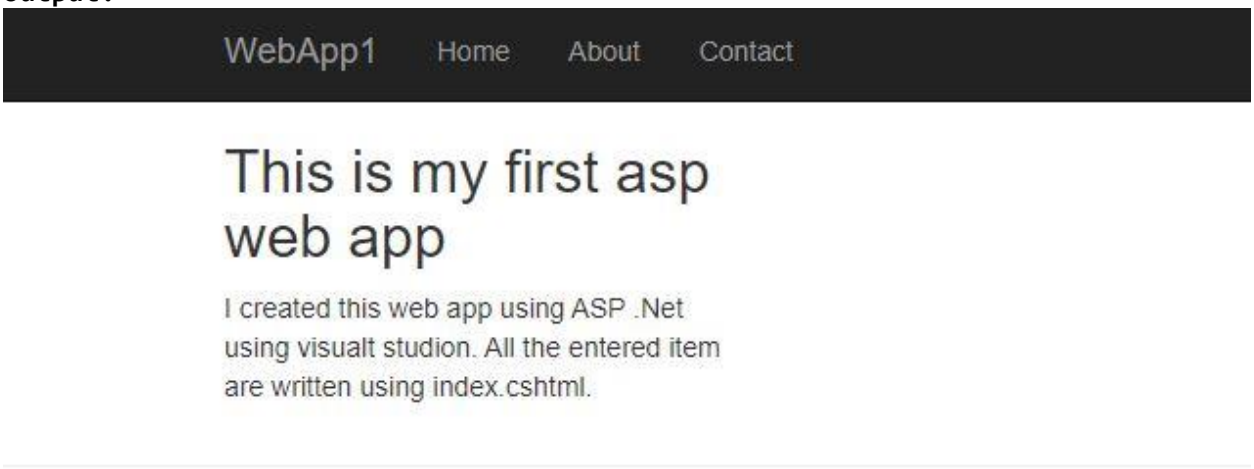
Index.cshtml

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
<div class="row">

    <div class="col-md-3">
        <h2>This is my first asp web app</h2>

        <p>I created this web app using ASP .Net using visualt studion. All
the entered item are written using index.cshtml.</p>
    </div>

    </div>
</div>
```

About.cshtml

```
@page
@model AboutModel
@{
    ViewData["Title"] = "About";
}
<h2>@ViewData["Title"]</h2>
<h3>@Model.Message</h3>

<p>Use this area to provide additional information.</p>
```

Output:



//MVC program to demonstrate model, view, controller
Department Controller

```
using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using EmployeeManagement.Models;

namespace EmployeeManagement.Controllers

{

   public class DepartmentController : Controller

   {

      public IActionResult Index()

      {


         var department = Department.GetDepartments();


         return View(department);

      }


      public ActionResult Detail(int id)

      {

         var departments = Department.GetDepartments();

         var emp = departments.FirstOrDefault(x => x.Id == id);

         return View(emp);


      }
```

```
        public ActionResult Add()
        {
            return View();
        }
        [HttpPost]
         public string Add(Department department)
        {
           return "Record Saved";
        }
    }
}
```

Department Model

```
using System.ComponentModel.DataAnnotations;
using System.Collections.Generic;
namespace EmployeeManagement.Models
{
    public class Department
    {
        public int Id {get; set;}

        [Required(ErrorMessage="Department name Required!!")]
        public string DepartmentName {get;  set;}

         public int DepartmentNo {get;  set;}

           public string Floor {get;  set;}
        public float Salary { get; set;}
```

```
public static List<Department> GetDepartments()
    {Department department1 = new Department()
    {
        Id = 103,
        DepartmentName = "Finance",
        DepartmentNo = 770,
        Floor = "3rd",
        Salary = 3000000
    };
    Department department2 = new Department()
    {
        Id = 104,
        DepartmentName = "HR",
        DepartmentNo = 800,
        Floor = "1st",
        Salary = 8900000
    };
    Department department3 = new Department()
    {
        Id = 105,
        DepartmentName = "Marketing",
         DepartmentNo = 900,
         Floor = "2nd",
        Salary = 8500000
    };
    List<Department> departments =  new List<Department>() {department1,
department2,department3};


    return departments;
```

```
        }
    }
}
```

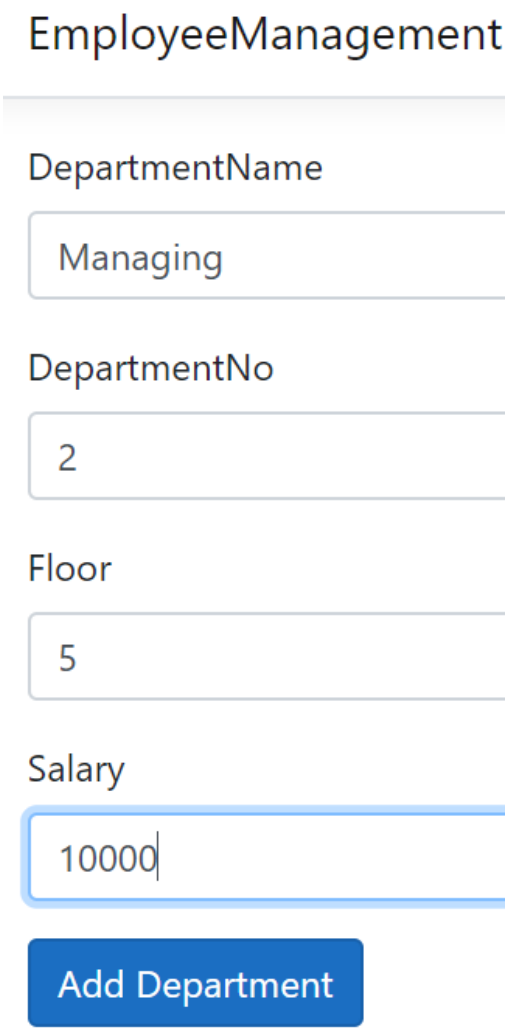
Department View

Add.cshtml

```
@model Department
@{
  ViewData["t"] = "Add Department";
}
<form asp-controller="Department" asp-action="Add" method="POST">
  <div class="form-group">
    <label asp-for="DepartmentName">DepartmentName</label>
    <span asp-validation-for="DepartmentName" class="text-danger"></span>
    <input type="text" class="form-control" asp-for="DepartmentName"  placeholder="Enter Department name">


  </div>
  <div class="form-group">
    <label asp-for="DepartmentNo">DepartmentNo</label>
    <input type="number" class="form-control" asp-for="DepartmentNo" placeholder="Enter Department No">
  </div>



      <div class="form-group">
        <label asp-for="Floor">Floor</label>
        <input type="number" class="form-control" asp-for="Floor" placeholder="Enter Floor Number">
      </div>
      <div class="form-group">
```

```
    <label asp-for="Salary">Salary</label>
    <input type="number" class="form-control" asp-for="Salary" placeholder="Enter Salary">
  </div>
    <button type="submit" class="btn btn-primary">Add Department</button>
</form>
```

Output:



Detail.cshtml

```
@model Department

<h2>@Model.DepartmentName Detail<a class="btn btn-primary" href="/Department/Index">Back to Department List</a>

</h2>

<ul class="list-group list-group-flush">

  <li class="list-group-item"></b>DepartmentName: @Model.DepartmentName</li>

   <li class="list-group-item"></b>DepartmentNo: @Model.DepartmentNo</li>

    <li class="list-group-item"></b>Floor: @Model.Floor</li>

  <li class="list-group-item"></b>Salary: @Model.Salary</li>

</ul>
```

Output: